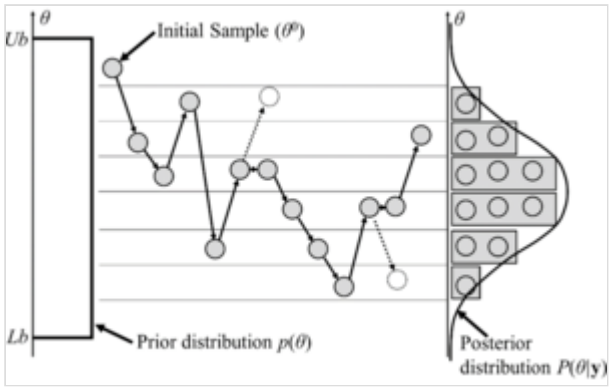
Metropolis–Hastings algorithm

In statistics and 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 from which direct sampling is difficult. New samples are added to the sequence in two steps: first a new sample is proposed based on the previous sample, then the proposed sample is either added to the sequence or rejected depending on the value of the probability distribution at that point. The resulting sequence can be used to approximate the distribution (e.g. to generate a histogram) or to compute an integral (e.g. an expected value).



The Metropolis-Hastings algorithm sampling a normal one-dimensional posterior probability distribution.

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, there are usually other methods (e.g. adaptive rejection sampling) that can directly return independent samples from the distribution, and these are free from the problem of autocorrelated samples that is inherent in MCMC methods.

History

The algorithm is named in part for Nicholas Metropolis, the first coauthor of a 1953 paper, entitled *Equation of State Calculations by Fast Computing Machines*, with Arianna W. Rosenbluth, Marshall Rosenbluth, Augusta H. Teller and Edward Teller. For many years the algorithm was known simply as the *Metropolis algorithm*.[1][2] The paper proposed the algorithm for the case of symmetrical proposal distributions, but in 1970, W.K. Hastings extended it to the more general case.[3] The generalized method was eventually identified by both names, although the first use of the term "Metropolis-Hastings algorithm" is unclear.

Some controversy exists with regard to credit for development of the Metropolis algorithm. Metropolis, who was familiar with the computational aspects of the method, had coined the term "Monte Carlo" in an earlier article with Stanisław Ulam, and led the group in the Theoretical Division that designed and built the MANIAC I computer used in the experiments in 1952. However, prior to 2003 there was no detailed account of the algorithm's development. Shortly before his death, Marshall Rosenbluth attended a 2003 conference at LANL marking the 50th anniversary of the 1953 publication. At this conference, Rosenbluth described the algorithm and its development in a presentation titled "Genesis of the Monte Carlo Algorithm for Statistical Mechanics".[4] Further historical clarification is made by Gubernatis in a 2005 journal article[5]

recounting the 50th anniversary conference. Rosenbluth makes it clear that he and his wife Arianna did the work, and that Metropolis played no role in the development other than providing computer time.

This contradicts an account by Edward Teller, who states in his memoirs that the five authors of the 1953 article worked together for "days (and nights)".[6] In contrast, the detailed account by Rosenbluth credits Teller with a crucial but early suggestion to "take advantage of statistical mechanics and take ensemble averages instead of following detailed kinematics". This, says Rosenbluth, started him thinking about the generalized Monte Carlo approach – a topic which he says he had discussed often with John Von Neumann. Arianna Rosenbluth recounted (to Gubernatis in 2003) that Augusta Teller started the computer work, but that Arianna herself took it over and wrote the code from scratch. In an oral history recorded shortly before his death,[7] Rosenbluth again credits Teller with posing the original problem, himself with solving it, and Arianna with programming the computer.

Description

The Metropolis–Hastings algorithm can draw samples from any probability distribution with probability density $P(x)$, provided that we know a function $f(x)$ proportional to the density P and the values of $f(x)$ can be calculated. The requirement that $f(x)$ must only be proportional to the density, rather than exactly equal to it, makes the Metropolis–Hastings algorithm particularly useful, because it removes the need to calculate the density's normalization factor, which is often extremely difficult in practice.

The Metropolis–Hastings algorithm generates 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. These sample values are produced iteratively in such a way, that the distribution of the next sample depends only on the current sample value, which makes the sequence of samples a Markov chain. Specifically, at each iteration, the algorithm proposes 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 it is rejected in which case the candidate value is discarded, and the 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.

The method used to propose new candidates is characterized by the probability distribution $g(x \mid y)$ (sometimes written $Q(x \mid y)$) of a new proposed sample x given the previous sample y. This is called the *proposal density*, *proposal function*, or *jumping distribution*. A common choice for $g(x \mid y)$ is 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 Gaussian random walk. In the original paper by Metropolis et al. (1953), $g(x \mid y)$ was suggested to be a uniform distribution limited to some maximum distance from y. More complicated proposal functions are also possible, such as those of Hamiltonian Monte Carlo, Langevin Monte Carlo, or preconditioned Crank–Nicolson.

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 density function $P(x)$ (a.k.a. a target distribution)[a].

1. Initialization: Choose an arbitrary point x_t to be the first observation in the sample and choose a proposal function $g(x \mid y)$. In this section, g is assumed to be symmetric; in other words, it must satisfy $g(x \mid y) = g(y \mid x)$.
2. For each iteration *t*:
   - *Propose* a candidate x' for the next sample by picking from the distribution $g(x' \mid x_t)$.
   - *Calculate* the *acceptance ratio* $\alpha = f(x')/f(x_t)$, which will be used to decide whether to accept or reject the candidate[b]. Because *f* is proportional to the density of *P*, we have that $\alpha = f(x')/f(x_t) = P(x')/P(x_t)$.
   - *Accept or reject*:
     - Generate a uniform random number $u \in [0, 1]$.
     - If $u \leq \alpha$, then *accept* the candidate by setting $x_{t+1} = x'$,
     - If $u > \alpha$, then *reject* the candidate and 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. $P(x)$ at specific point x is proportional to the iterations spent on the point by the algorithm. Note that the acceptance ratio α indicates how probable the new proposed sample is with respect to the current sample, according to the distribution whose density is $P(x)$. 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 $P(x)$ corresponding to an $\alpha > 1 \geq u$), 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 larger 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 with density $P(x)$.

Compared with an algorithm like adaptive rejection sampling[8] that directly generates independent samples from a distribution, Metropolis–Hastings and other MCMC algorithms have a number of disadvantages:

- The samples are autocorrelated. 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 effective sample sizes can be significantly lower than the number of samples actually taken, leading to large errors.
- 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,[9] where an initial number of samples 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 suitable 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. That way, the problem of sampling from potentially high-dimensional space will be reduced to a collection of problems to sample from small dimensionality.[10] This is especially applicable when the multivariate distribution is composed 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 the adaptive rejection sampling methods,[8] the adaptive rejection Metropolis sampling algorithm,[11] a simple one-dimensional Metropolis–Hastings step, or slice sampling.

Formal derivation

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 $\pi(x)$ such that $\pi(x) = P(x)$.[12]

A Markov process is uniquely defined by its transition probabilities $P(x' \mid x)$, the probability of transitioning from any given state x to any other given state x'. It has a unique stationary distribution $\pi(x)$ when the following two conditions are met:[12]

1. *Existence of stationary distribution*: there must exist a stationary distribution $\pi(x)$. A sufficient but not necessary condition is detailed balance, which requires that each transition $x \to x'$ 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, $\pi(x)P(x' \mid x) = \pi(x')P(x \mid x')$.
2. *Uniqueness of stationary distribution*: the stationary distribution $\pi(x)$ 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.

The Metropolis–Hastings algorithm involves designing a Markov process (by constructing transition probabilities) that fulfills the two above conditions, such that its stationary distribution $\pi(x)$ is chosen to be $P(x)$. The derivation of the algorithm starts with the condition of detailed balance:

$$P(x' \mid x)P(x) = P(x \mid x')P(x'),$$

which is re-written as

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

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

$$P(x' \mid x) = g(x' \mid x)A(x', x).$$

Inserting this relation in the previous equation, we have

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

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

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

For this Metropolis acceptance ratio A, either $A(x', x) = 1$ or $A(x, x') = 1$ and, either way, the condition is satisfied.

The Metropolis–Hastings algorithm can thus be written as follows:

1. Initialise
    1. Pick an initial state x_0.
    2. Set $t = 0$.
2. Iterate
    1. *Generate* a random candidate state x' according to $g(x' \mid x_t)$.
    2. *Calculate* the acceptance probability $A(x', x_t) = \min\left(1, \frac{P(x')}{P(x_t)} \frac{g(x_t \mid x')}{g(x' \mid x_t)}\right)$.
    3. *Accept or reject*:
        1. generate a uniform random number $u \in [0, 1]$;
        2. if $u \leq A(x', x_t)$, then *accept* the new state and set $x_{t+1} = x'$;
        3. if $u > A(x', x_t)$, then *reject* the new state, and copy the old state forward $x_{t+1} = x_t$.
    4. *Increment*: set $t = t + 1$.

Provided that specified conditions are met, the empirical distribution of saved states $x_0, \ldots, x_T$ will approach $P(x)$. The number of iterations (T) required to effectively estimate $P(x)$ depends on the number of factors, including the relationship between $P(x)$ and the proposal distribution and the desired accuracy of estimation.[13] For distribution on discrete state spaces, it has to be of the order of the autocorrelation time of the Markov process.[14]

It is important to notice that it is not clear, in a general problem, which distribution $g(x' \mid x)$ one should use or the number of iterations necessary for proper estimation; both are free parameters of the method, which must be adjusted to the particular problem in hand.

Use in numerical integration

A common use of Metropolis–Hastings algorithm is to compute an integral. Specifically, consider a space $\Omega \subset \mathbb{R}$ and a probability distribution $P(x)$ over Ω, $x \in \Omega$. Metropolis–Hastings can estimate an integral of the form of

$$P(E) = \int_{\Omega} A(x) P(x)\, dx,$$

where $A(x)$ is a (measurable) function of interest.

For example, consider a statistic $E(x)$ and its probability distribution $P(E)$, which is a marginal distribution. Suppose that the goal is to estimate $P(E)$ for E on the tail of $P(E)$. Formally, $P(E)$ can be written as

$$P(E) = \int_{\Omega} P(E \mid x) P(x)\, dx = \int_{\Omega} \delta\big(E - E(x)\big) P(x)\, dx = E\big(P(E \mid X)\big)$$

and, thus, estimating $P(E)$ can be accomplished by estimating the expected value of the indicator function $A_E(x) \equiv \mathbf{1}_E(x)$, which is 1 when $E(x) \in [E, E + \Delta E]$ and zero otherwise. Because E is on the tail of $P(E)$, the probability to draw a state x with $E(x)$ on the tail of $P(E)$ is proportional to $P(E)$, which is small by definition. The Metropolis–Hastings algorithm can be used here to sample (rare) states more likely and thus increase the number of samples used to estimate $P(E)$ on the tails. This can be done e.g. by using a sampling distribution $\pi(x)$ to favor those states (e.g. $\pi(x) \propto e^{aE}$ with $a > 0$).

Step-by-step instructions

Suppose that 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 $g(x' \mid x_t)$ 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

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

is the ratio of the proposal density in two directions (from x_t to x' and conversely). 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.

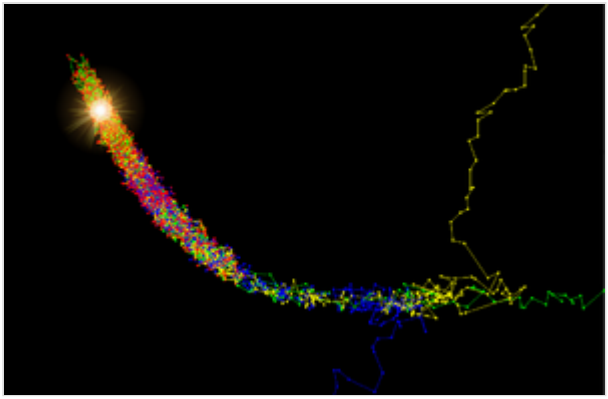
If $a \geq 1$:

$$x_{t+1} = x',$$

else:

$$x_{t+1} = \begin{cases} x' & \text{with probability } a, \\ x_t & \text{with probability } 1 - a. \end{cases}$$

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 $P(x)$.



Three Markov chains running on the 3D Rosenbrock function using the Metropolis–Hastings algorithm. The chains converge and mix in the region where the function is high. The approximate position of the maximum has been illuminated. The red points are the ones that remain after the burn-in process. The earlier ones have been discarded.

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 $g(x' \mid x_t) \approx P(x')$. If a Gaussian proposal density g 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 about 50%, decreasing to about 23% for an N-dimensional Gaussian target distribution.[15] These guidelines can work well when sampling from sufficiently regular Bayesian posteriors as they often follow a multivariate normal distribution as can be established using the Bernstein–von Mises theorem.[16]

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. One typically tunes the proposal distribution so that the algorithms accepts on the order of 30% of all samples – in line with the theoretical estimates mentioned in the previous paragraph.

Bayesian Inference

MCMC can be used to draw samples from the posterior distribution of a statistical model. The acceptance probability is given by: $P_{acc}(\theta_i \to \theta^*) = \min\left(1, \frac{\mathcal{L}(y|\theta^*)P(\theta^*)}{\mathcal{L}(y|\theta_i)P(\theta_i)} \frac{Q(\theta_i|\theta^*)}{Q(\theta^*|\theta_i)}\right)$, where $\mathcal{L}$ is the likelihood, $P(\theta)$ the prior probability density and Q the (conditional) proposal probability.

See also

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

References

1. Kalos, Malvin H.; Whitlock, Paula A. (1986). *Monte Carlo Methods Volume I: Basics*. New York: Wiley. pp. 78–88.
2. Tierney, Luke (1994). "Markov chains for exploring posterior distributions" (https://projecteuclid.org/journals/annals-of-statistics/volume-22/issue-4/Markov-Chains-for-Exploring-Posterior-Distributions/10.1214/aos/1176325750.full). *The Annals of Statistics*. **22** (4): 1701–1762. doi:10.1214/aos/1176325750 (https://doi.org/10.1214%2Faos%2F1176325750).
3. Hastings, W.K. (1970). "Monte Carlo Sampling Methods Using Markov Chains and Their Applications". *Biometrika*. **57** (1): 97–109. Bibcode:1970Bimka..57...97H (https://ui.adsabs.harvard.edu/abs/1970Bimka..57...97H). doi:10.1093/biomet/57.1.97 (https://doi.org/10.1093%2Fbiomet%2F57.1.97). JSTOR 2334940 (https://www.jstor.org/stable/2334940). Zbl 0219.65008 (https://zbmath.org/?format=complete&q=an:0219.65008).
4. M.N. Rosenbluth (2003). "Genesis of the Monte Carlo Algorithm for Statistical Mechanics". *AIP Conference Proceedings*. **690**: 22–30. Bibcode:2003AIPC..690...22R (https://ui.adsabs.harvard.edu/abs/2003AIPC..690...22R). doi:10.1063/1.1632112 (https://doi.org/10.1063%2F1.1632112).
5. J.E. Gubernatis (2005). "Marshall Rosenbluth and the Metropolis Algorithm" (https://zenodo.org/record/1231899). *Physics of Plasmas*. **12** (5): 057303. Bibcode:2005PhPl...12e7303G (https://ui.adsabs.harvard.edu/abs/2005PhPl...12e7303G). doi:10.1063/1.1887186 (https://doi.org/10.1063%2F1.1887186).
6. Teller, Edward. *Memoirs: A Twentieth-Century Journey in Science and Politics*. Perseus Publishing, 2001, p. 328
7. Rosenbluth, Marshall. "Oral History Transcript" (https://www.aip.org/history-programs/niels-bohr-library/oral-histories/28636-1). American Institute of Physics
8. Gilks, W. R.; Wild, P. (1992-01-01). "Adaptive Rejection Sampling for Gibbs Sampling". *Journal of the Royal Statistical Society. Series C (Applied Statistics)*. **41** (2): 337–348. doi:10.2307/2347565 (https://doi.org/10.2307%2F2347565). JSTOR 2347565 (https://www.jstor.org/stable/2347565).
9. *Bayesian data analysis*. Gelman, Andrew (2nd ed.). Boca Raton, Fla.: Chapman & Hall / CRC. 2004. ISBN 978-1584883883. OCLC 51991499 (https://search.worldcat.org/oclc/51991499).

10. Lee, Se Yoon (2021). "Gibbs sampler and coordinate ascent variational inference: A set-theoretical review". *Communications in Statistics - Theory and Methods*. **51** (6): 1549–1568. arXiv:2008.01006 (https://arxiv.org/abs/2008.01006). doi:10.1080/03610926.2021.1921214 (https://doi.org/10.1080%2F03610926.2021.1921214). S2CID 220935477 (https://api.semanticscholar.org/CorpusID:220935477).
11. Gilks, W. R.; Best, N. G.; Tan, K. K. C. (1995-01-01). "Adaptive Rejection Metropolis Sampling within Gibbs Sampling". *Journal of the Royal Statistical Society. Series C (Applied Statistics)*. **44** (4): 455–472. doi:10.2307/2986138 (https://doi.org/10.2307%2F2986138). JSTOR 2986138 (https://www.jstor.org/stable/2986138).
12. Robert, Christian; Casella, George (2004). *Monte Carlo Statistical Methods* (https://archive.org/details/springer_10.1007-978-1-4757-4145-2). Springer. ISBN 978-0387212395.
13. Raftery, Adrian E., and Steven Lewis. "How Many Iterations in the Gibbs Sampler?" *In Bayesian Statistics 4*. 1992.
14. Newman, M. E. J.; Barkema, G. T. (1999). *Monte Carlo Methods in Statistical Physics*. USA: Oxford University Press. ISBN 978-0198517979.
15. Roberts, G.O.; Gelman, A.; Gilks, W.R. (1997). "Weak convergence and optimal scaling of random walk Metropolis algorithms" (http://www.stat.columbia.edu/~gelman/research/published/theory7.ps). *Ann. Appl. Probab.* **7** (1): 110–120. CiteSeerX 10.1.1.717.2582 (https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.717.2582). doi:10.1214/aoap/1034625254 (https://doi.org/10.1214%2Faoap%2F1034625254).
16. Schmon, Sebastian M.; Gagnon, Philippe (2022-04-15). "Optimal scaling of random walk Metropolis algorithms using Bayesian large-sample asymptotics" (https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8924149). *Statistics and Computing*. **32** (2): 28. doi:10.1007/s11222-022-10080-8 (https://doi.org/10.1007%2Fs11222-022-10080-8). ISSN 0960-3174 (https://search.worldcat.org/issn/0960-3174). PMC 8924149 (https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8924149). PMID 35310543 (https://pubmed.ncbi.nlm.nih.gov/35310543).

Notes

a. In the original paper by Metropolis et al. (1953), f was taken to be the Boltzmann distribution as the specific application considered was Monte Carlo integration of equations of state in physical chemistry; the extension by Hastings generalized to an arbitrary distribution f.

b. In the original paper by Metropolis et al. (1953), f was actually the Boltzmann distribution, as it was applied to physical systems in the context of statistical mechanics (e.g., a maximal-entropy distribution of microstates for a given temperature at thermal equilibrium). Consequently, the acceptance ratio was itself an exponential of the difference in the parameters of the numerator and denominator of this ratio.

Further reading

- Bernd A. Berg. *Markov Chain Monte Carlo Simulations and Their Statistical Analysis*. Singapore, World Scientific, 2004.
- Chib, Siddhartha; Greenberg, Edward (1995). "Understanding the Metropolis–Hastings Algorithm" (https://www.jstor.org/stable/2684568). *The American Statistician*, 49(4), 327–335.
- David D. L. Minh and Do Le Minh. "Understanding the Hastings Algorithm." Communications in Statistics - Simulation and Computation, 44:2 332–349, 2015 (http://www.tandfonline.com/doi/abs/10.1080/03610918.2013.777455#.VOk8J1PF9_c)
- Bolstad, William M. (2010) *Understanding Computational Bayesian Statistics*, John Wiley & Sons ISBN 0-470-04609-0