International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 07 Issue: 07 | July - 2023 SJIF Rating: 8.176 ISSN: 2582-3930

Generalization of Metropolis – Hasting Algorithm

Dr. Suneel Pappala,
Associate Professor,
Information Technology, Lords Institute of Engineering & Technology,
Osmania University, Hyderabad, Telangana, INDIA.

Abstract: This paper presents Metropolis to search the space of possible configurations, by exploring possible transitions between configurations. The way the Metropolis algorithm decides about moving from the current state to a state in the neighborhood can be seen as a two-stage process. a probability that depends upon the relative costs of the solutions associated with two states. The Metropolis algorithm as the proposal stage will choose a neighbor uniformly at random. the Hasting Algorithm chain has the same stationary distribution as the usual Metropolis algorithm.

Keywords: *Metropolis Algorithm, Hasting Algorithm, Boltzmann Factor, Random Search.*

Metropolis Algorithm (MA)

The Metropolis Algorithm (MA) is based on the notion of complete balance that describes equilibrium for systems whose configurations have probability proportional to the Boltzmann factor. The Boltzmann factor, e-m, is proportional to the probability that the system will be found in a particular configuration at energy E when the temperature of the environment is T. That is,

$$P \propto e^{-E/T}$$

Our objective is to search the space of possible configuration, by exploring possible transitions between configurations. We consider two configurations A and B where each of which occurs with probabilities propositional to Boltzmann factor. A comparative probability P = P(A)/P(B) is used to decide which configuration to accept according to the following algorithm;

1. Starting from a configuration A, with known energy E_A, make a small change in the A to obtain a new configuration B.

2. Compute Ea for a configuration B.

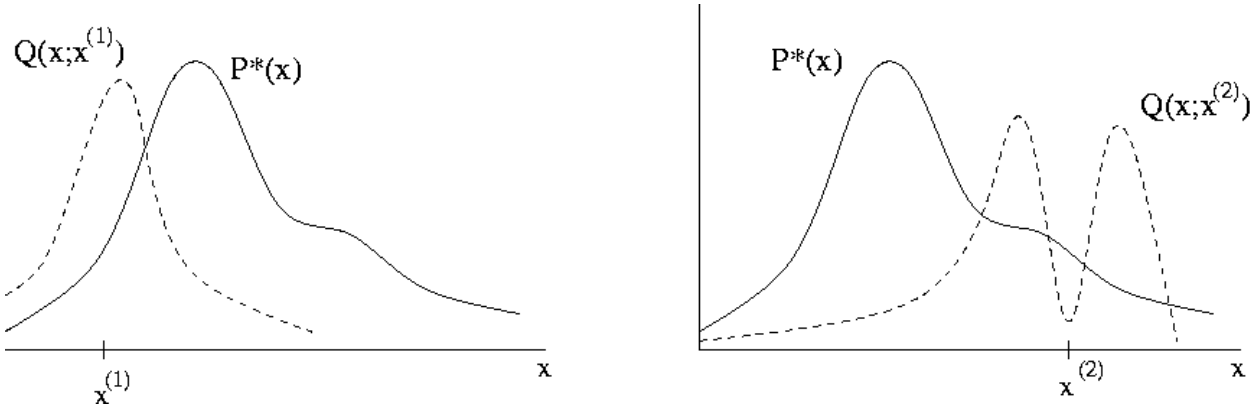
3. If $E_a < E_A$, accept the new configuration, since it has lower energy (a desirable thing, according to the Boltzmann factor).

4. If $E_B > E_A$, accept the new configuration with probability p where.

$$p = e^{-(E_A - E_B)/T}$$

This means that when the temperature is high, we don't mind taking steps in the "wrong direction, but as the temperature is lowered, we are bound to settle into the lowest configuration. If we follow these rules, then we will sample points in the space of all possible configurations with probability proportional to the Boltzmann factor, consistent with the theory of equilibrium statistical mechanics.

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. This sequence can be used to approximate the distribution (e.g. to generate a histogram) or to compute an integral (e.g. 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, 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.



The Metropolis–Hastings (MH) algorithm (Metropolis et al., 1953; Hastings, 1970) is the most popular technique to build Markov Chains with a given invariant distribution (see, e.g., Gillespie, 1992; Tierney, 1994; Gilks et al., 1995; Gamerman, 1997; Robert and Casella, 1999).

The general MH algorithm considers two distributions, namely the target distribution σ and the proposal conditional distribution $q(\mathbf{x}^*|\mathbf{x})$ from which a candidate sample $\mathbf{x}^*$ for the new Markov chain state is drawn. If

the current state of the chain is **x**, then, according to the MH algorithm, the chain moves to its new state $\mathbf{x}^*$ with the probability

$$P(\mathrm{x}, \mathrm{x}^*) = min\left[1, \frac{\sigma(\mathrm{x}^*)\, q(\mathrm{x}|\mathrm{x}^*)}{\sigma(\mathrm{x})\, q(\mathrm{x}^*|\mathrm{x})}\right],$$

or remains at **x**.

The pseudo-code for the MH algorithm is shows three sets of samples generated by the MH algorithm for 1D multimodal distribution σ(**x**) assuming that $\mathbf{x}^*$ are generated from the uniform distribution $q(\mathbf{x}^*|\mathbf{x}) = const$.

- Initialize $\mathbf{x}^0$
- Repeat
  - generate uniform random number $u \sim U(0,1)$
  - generate test sample $\mathbf{x}^* \sim q(\mathbf{x}^*|\mathbf{x}^i)$
  - if $u < P(\mathbf{x}^i, \mathbf{x}^*) = \min\left[1, \frac{\sigma(\mathbf{x}^*)q(\mathbf{x}^i|\mathbf{x}^*)}{\sigma(\mathbf{x}^i)q(\mathbf{x}^*|\mathbf{x}^i)}\right]$
    $\mathbf{x}^{i+1} = \mathbf{x}^*$
    otherwise
    $\mathbf{x}^{i+1} = \mathbf{x}^i$
- Continue until sufficient number of samples $\{\mathbf{x}^i\}$ are generated

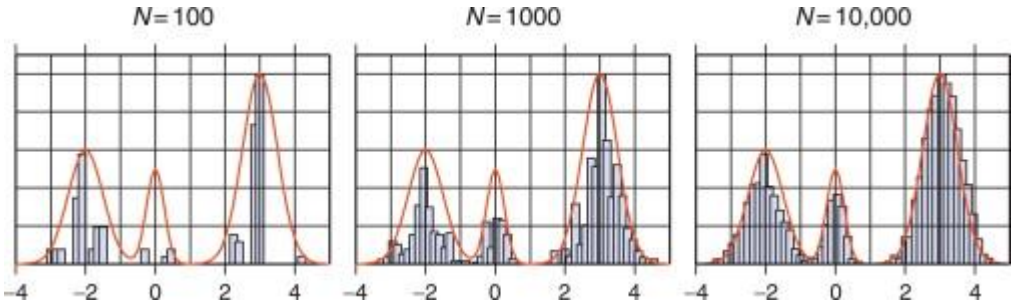


Fig. Histograms of samples generated by the Metropolis–Hastings algorithm from the target distribution

Note that the MH algorithm does not require knowledge of the absolute values of σ(**x**) but only its ratio for the current and the proposed states. Thus, σ(**x**) can be defined up to a constant factor. In particular, it does not need to be normalizable.

The transition kernel for the MH algorithm reads

$$K_{MH}\left(x^{i+1},x^{i}\right) = q\left(x^{i+1}\middle|x^{i}\right)P\left(x^{i},x^{i+1}\right) + \delta\left(x^{i}-x^{i+1}\right)R\left(x^{i}\right)$$

Where the second term takes into account that the chain can reject a proposed $\mathbf{x} \rightarrow \mathbf{x}^{*}$ movement and stay in the same state with the probability

$$R\left(x^{i}\right) = \int q\left(x^{*}\middle|x^{i}\right)\left(1-P\left(x^{i},x^{*}\right)dx^{*}\right)$$

It is easy to check that by construction of $\mathbf{K}_{MH}$, the equilibrium condition of Eq. (140) is fulfilled for $\sigma(\mathbf{x}^{i})$:

$$\sigma\left(x^{i}\right)K_{MH}\left(x^{i+1},x^{i}\right) = \sigma\left(x^{i+1}\right)K_{MH}\left(x^{i},x^{i+1}\right)$$

which allows $\sigma(\mathbf{x})$ to be an invariant distribution. However, to show that the MH algorithm converges to $\pi \equiv \sigma$, it is necessary to show that the chain generated according to the algorithm is an ergodic one. The aperiodicity of the chain follows from the fact that it always allows the proposal of the move to be rejected. To ensure irreducibility, we need to make sure that the support of $q(\mathbf{x})$ includes support of $\sigma(\mathbf{x})$. Tierney (1994) has shown that in such a case $\sigma(\mathbf{x})$ is the invariant distribution of the chain.

Depending on the choice of the auxiliary distribution $q(\mathbf{x})$, the MH algorithms can be classified into a few categories. Let us consider two, the most popular classes of MH algorithms.

The Algorithm

Let f(x) be the probability density (or probability mass) function of the distribution from which we wish to extract a sample of draws. We call f(x) the target distribution**.**

Denote by $q(x|x')$ a family of conditional distributions of our choice, from which it is easy to generate draws.

The vectors x,x' the argument of f(x) all have the same dimension.

The Metropolis-Hastings algorithm starts from any value x_1 belonging to the support of the target distribution.

The value x_1 can be user-defined or extracted from a given distribution.

Then, the subsequent values $x_2,x_3.....x_t$ are generated recursively.

In particular, the value f(x) at time step x_1 is generated as follows:

1. draw y_t from the distribution with density $q(y_t|x_{t-1})$
2. set

$$p_t = \min\left(\frac{f(y_t)}{f(x_{t-1})}\frac{q(x_{t-1}|y_t)}{q(y_t|x_{t-1})}, 1\right)$$

3. draw u_t from a uniform distribution on [0,1];

   if $u_t \leq p_t$, set $x_t = y_t$; otherwise, set $x_t = x_{t-1}$.

   Since u_t is Uniform

   $$P(u_t \leq p_t) = p_t$$

   that is, the probability of accepting the proposal y_t as the new draw x_t is equal to p_t.

Hasting's Generalization of Metropolis Algorithm:

The way the Metropolis algorithm decides about moving from the current state s_i to a state in the neighborhood can be seen as a two-stage process: first, choose a neighbor s_j uniformly at random (the proposal stage), and then, with a probability α which depends upon the relative costs of the solutions associated with s_j and s_i, move to s_j or remain at s_i (the acceptance stage).

In our case, the neighbors of a state [A|I] are [A' |I]'s where A' is obtained by performing an elementary operation using the vectors in A and I. Some of the elementary operations represent what we call long jumps because a vector v is replaced by another u where there is a large difference in the norms of v and u. This happens when v is replaced by v ± cw when the constant c is large. It is desirable to have a control on how extensively our algorithm will make use of such long jumps. This is not possible in the

Algorithm Metropolis Algorithm

1. Input: B ← Basis for the lattice L and a rational number K
2. Output: Matrix R' such that B ∗ R' contains a vector v with ||v|| ≤ K.
3. Let I ← n × n Identity matrix. Let R' = [R|I] be the starting state in the search space as in Definition and C (R') denote cost of R' as defined in the beginning of this section.
4. Set BestNorm = C (R')
5. while BestNorm > K do
6. Select any one of the neighbor S' of R' uniformly at random by performing one of the elementary operations as defined

7. if BestNorm > C (S') then
8. BestNorm = C (S')
9. end if
10. Set R'= S' with probability

$$\alpha = \min\left(\frac{e^{-c(S')/T}}{e^{-c(R')/T}}, 1\right)$$

11. end while

Metropolis algorithm as the proposal stage will chose a neighbor uniformly at random. To overcome this problem, we make use of the Hasting's generalization of the Metropolis algorithm. In this generalization, we can use any probability to select the neighbor of a state in the proposal stage. Let q_{xz} denote the probability by which we select a neighbor z when the current state is x. Let x be a state. If $y_1, \ldots, y_{nx}$ be neighbors the neighbors of x. Then

$$q_{xz} = \begin{cases} 0 & \text{if } x \neq z \text{ and } z \notin N(x) \\ \theta & \text{if } x = z, \\ r_i & \text{if } z = y_i \end{cases}$$

where the values r_i can be chosen appropriately depending on how much we want to invest on each of the strategy. The Hasting's generalized metropolis algorithm M2 runs on the same state space but has a different transition probability: Suppose the chain M2 is at a state the state x at some step. Then

1) With probability q_{xz}, M_2 selects a state z in the neighborhood.

2) If z = x then the next state of M_2 is x.

3) If $z = y_i$, we first compute α defined as

$$\alpha = \min\left(\frac{e^{-c(y_i)/T} \cdot q_{y_i x}}{e^{-c(x)}/T \cdot q_{x y_i}}, 1\right)$$

Here, for any state z, c(z) represents the cost of the candidate solution of z and T is a fixed temperature parameter.

4) We move to y_i with probability α else we remain in the present state x.

It can be verified easily that the chain M2 is time-reversible and the in its stationary distribution, the probability of x, πx is given by:

$$\pi_x = \frac{e^{-c(x)/T}}{Z},$$

where Z is the normalizing factor $\sum_i \pi_i$. The chain M_2 is the Hasting's generalization. This chain has the same stationary distribution as the usual Metropolis algorithm, but has the flexibility of fine tuning the probability of choosing a neighbor to reflect the structure of the problem at hand. In our implementation, we shall keep q_{xyi} the same as q_{yix}. The detailed algorithm (Algorithm 2) is given below. In the next section we will compare the results of our algorithm with that of LLL algorithm.

Algorithm 2 Hasting's Generalization

1: Input: B ← Basis for the lattice L and a rational number K

2: Output: Matrix R 0 such that B ∗ R' contains a vector v with ||v|| ≤ K.

3: Let I ← n × n Identity matrix. Let R' = [R|I] be the starting state in the search space as in Definition and C(R 0) denote cost of R' as defined in the beginning of this section. Let d denote total number of neighbors

4: Set BestNorm = C(R')

5: while BestNorm > K do

6: Select any one of the neighbor S' of R' by performing one of the elementary operations defined below.

- Swap two columns of R with probability ${}^{m}C_2/ d$,
- Multiply a column of R by −1 with probability m/d
- Add a power of 2 times a column of R0 to another column of R i.e. in particular, ri ← r 0 i±c×r 0 j ,(i 6= j, 1 ≤ i ≤ m, 1 ≤ j ≤ m + n, where C= 20 , . . . , 2 k , k = n · log (αn)) with probability d−mC2−m d · Pi ,

where Pi denote probability of selecting the value of C = 2i and Pk i=0 Pi = 1.

[We can use more than one probability distribution to select values for c. In our implementation we have selected two probability distributions Pi = 1 k+1 and Qi = 2(k + 1−i)/(k + 1)(k + 2) to select values for c. We will keep on changing our selection probability distribution with Pi and Qi for every selected number of steps (500 steps).]

7: if BestNorm > C(S') then

8: BestNorm = C(S')

9: end if

10: Set R'= S' with probability.

$$\alpha = \min\left(\frac{e^{-c(S')/T}}{e^{-c(R')/T}}, 1\right)$$

11.End while

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).

A Markov process is uniquely defined by its transition probabilities P(x'|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:

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, π(x)P(x'|x)= π(x') P(x|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.

The Metropolis–Hasting's algorithm involves designing a Markov process (by constructing transition probabilities) that fulfills 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:

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

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'/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) or A (x, x') and, either way, the condition is satisfied. The Metropolis–Hasting's algorithm can thus be written as follows:

1. Initialize
    1. Pick an initial state x_0.
    2. Set t=0.
2. Iterate
    1. *Generate* a random candidate state x' according to g (x'/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 € [0,1];
        2. if u<= 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 .
    4. *Increment*: set t=t+1.

Provided that specified conditions are met, the empirical distribution of saved states x_0, x_1, …… 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. For distribution on discrete state spaces, it has to be of the order of the autocorrelation time of the Markov process.

It is important to notice that it is not clear, in a general problem, which distribution g(x'/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.

REFERENCES:

[1] S. Sanyal, S. Raja, and S. Biswas, "Necessary and sufficient conditions for success of the metropolis algorithm for optimization," in GECCO'10, Copyright ACM, Portland, USA, 2010.

[2] T. Carson, "Emperical and analytic approaches to understanding local search heuristics," in PhD Thesis, University of California, San Diego, 2001

[3] P. V. E. Boas, "Another np-complete problem and the complexity of computing short vector in a lattice," in Tech. rep 8104, University of Amsterdam, Department of Mathematics, Netherlands, 1981.

[4] M. Ajtai, "The shortest vector problem in l2 is np-hard for randomized reductions," in STOC 98: Proceedings of the 30th Annual ACM Symposium on Theory of Computing, New York, NY, USA, 1998, pp. 10–19.

[5] D. Micciancio, "The shortest vector in a lattice is hard to approximate to within some constant," SIAM Journal of Computing, vol. 30(6), pp. 2008–2035, 2001.

[6] A. Lenstra, H. W. L. Jr., and L. Lovasz, "Factoring polynomials with rational coefficients," Mathematische Annalen, vol. 261(4), pp. 515–534, 1982.

[7] C. Schnorr, "A more efficient algorithm for lattice basis reduction," Journal of Algorithms, vol. 9(1), pp. 47–62, 1988.

[8] M. Ajtai, "The worst-case behavior of schnorr's algorithm approximating the shortest nonzero vector in a lattice," in STOC-03: Proceedings of the 35th Annual ACM Symposium on Theory of Computing, ACM Press, 2003, pp. 396–406.

[9] M. Mitzenmacher and E. Upfal, Probability and Computing: Randomized Algorithms and Probabilistic Analysis. Page 269: Cambridge University Press, 2005.

[10] S. reference v4.7, "Cryptography," www.sagemath.org/doc/reference/ sage/crypto/lattice.html.

[11] M. Ajtai, "Generating hard instances of lattice problems (extended abstract)," in Proceedings of the twenty-eighth annual ACM symposium on Theory of computing, STOC'96, ACM, New York, NY, USA, 1996.

[12] J. Buchmann, R. Lindner, M. Ruckert, and M. schneider, "Explicit hard instances of the shortest vector problem," in PQ Crypto, 2 nd Internation Workshop on Post Quantum Cryptography, LNCS 5299, 2008, pp. 79– 94.

[13] T. Darmstadt, "Lattice challenge," www.latticechallenge.orgS