




An Overview of MCMC Methods: From Theory to Applications

Christos Karras¹ , Aristeidis Karras¹ , Markos Avlonitis² ,
and Spyros Sioutas¹ 

¹ Computer Engineering and Informatics Department, University of Patras,
26504 Patras, Greece

{c.karras, akarras, sioutas}@ceid.upatras.gr

² Department of Informatics, Ionian University, 49100 Corfu, Greece
avlon@ionio.gr

Abstract. Markov Chain Monte Carlo techniques are used to generate samples that closely approximate a given multivariate probability distribution, with the function not having to be normalised in the case of certain algorithms such as Metropolis-Hastings. As with other Monte Carlo techniques, MCMC employs repeated random sampling to exploit the law of large numbers. Samples are generated by running a Markov Chain, which is created such that its stationary distribution follows the input function, for which a proposal distribution is used. This approach may be used for optimization tasks, for approximating solutions to non-deterministic polynomial time problems, for estimating integrals using importance sampling, and for cryptographic decoding. This paper serves as an introduction to the MCMC techniques and some of its applications.

Keywords: Metropolis-hastings · Markov Chains · Monte Carlo · MCMC methods · Gibbs sampling · Rejection sampling · Bayesian statistics

1 Introduction

Sampling across distributions is a significant concept in statistics, probability, systems engineering, and other fields that make use of stochastic models ([2, 8, 11, 20, 21]). Although sampling has long history, modern methods such as event detection and pattern recognition often rely to reservoir sampling methods as in [15] whereabouts the elements derived from a data stream are placed within a reservoir for further processing. Sampling from a multidimensional distribution is required for a variety of purposes, most notably to estimate sums and to approximate integrals that are highly insoluble analytically. However, typical sampling techniques such as rejection sampling are inadequate for this task, since they do not scale well with increasing dimensions, as the state space expands exponentially and hence rejection rates increase. *Markov Chain Monte Carlo*

(MCMC) techniques may be employed, given that their advanced variants are ideally suited for sampling from high-dimensional space. A brief introduction to MCMC methods is provided here to familiarise readers with the concept.

Moreover, an introduction to the mathematical underpinnings of *Markov Chains* is provided in order to aid in the comprehension of MCMC techniques. Following that, the fundamental concept behind *Monte Carlo* techniques is described. Based on this foundation, these two fundamental principles may be merged to form MCMC, with a particular emphasis on the popular *Metropolis-Hastings* algorithm (MH) [4] and the specific case of *Gibbs sampling* [7]. Following that, various elements such as parameter adjustment and convergence measurement are covered.

Finally, applications of these techniques are shown and described in further detail. It is feasible to decrypt encrypted documents [3], optimize functions [17], estimate integrals using generalized linear mixed models [22], and discover approximate solutions to non-deterministic polynomial-time (NP) hard problems using the Metropolis-Hastings algorithm [23].

2 Markov Chains

Stochastic processes are a series of random variables $(X_t)_{t \in T}$, that describe the states of a potentially infinitely vast state space S of a system at various points in time t . Only discrete time steps are considered in this case, hence $t \in \mathbb{N}_0$.

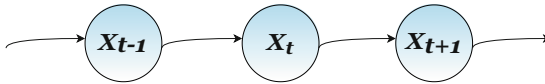


Fig. 1. A random variable X_t only depends on its immediate predecessor X_{t-1} , not on any others. With this Markov property, the structure of the dependence graph resembles a chain, hence the name Markov Chain.

Markov Chains are used to simulate stochastic processes in which the state of the next time step is determined by a small number of prior time steps. Only the final one is relevant in this paper, as seen in Fig. 1. The manner in which that condition was attained has no bearing on the subsequent state. This lack of memory in stochastic systems is often referred to as the Markov property:

$$\forall t \in \mathbb{N}_0, \forall i, j, k, \dots \in S :$$

$$\Pr [X_{t+1} = j \mid X_t = i, X_{t-1} = k, \dots] = \Pr [X_{t+1} = j \mid X_t = i] \quad (1)$$

A Markov Chain is composed of a collection of states S , a start distribution $q^{(0)}$, and the accompanying transition probabilities $p_{i,j} = \Pr [X_{t+1} = j \mid X_t = i]$

from one state to the next at a time step of t . Take note that transition probabilities are independent of time t in this case, implying that temporal invariance is true:

$$\forall t, t' \in \mathbb{N}_0 : \Pr [X_{t+1} = j \mid X_t = i] = \Pr [X_{t'+1} = j \mid X_{t'} = i] \quad (2)$$

At each time step t , the stochastic process may be in just one state, which does not have to be known. Thus, the distribution $q^{(t)}$ is introduced as a measure of probability for states, where $q_i^{(t)}$ is the likelihood that the stochastic process is in state $i \in S$ at time step t :

$$q_i^{(t)} = \Pr [X_t = i] \quad (3)$$

Thus, to satisfy the conditions for a probability distribution, $\forall i \in S, \forall t \in \mathbb{N}_0 : q_i^{(t)} \geq 0$ and $\sum_{i \in S} q_i^{(t)} = 1$, or $\int_{i \in S} q_i^{(t)} di = 1$, respectively. For the sake of simplicity, the following computations will disregard the scenario when S is not finite. Analogously, the continuous case follows.

Thus, assuming S is finite, the transition probabilities between states i and j may be expressed as a matrix $P := (p_{i,j})$ and the distributions as vectors $q^{(t)} = (q_1^{(t)} \dots q_n^{(t)}) \in \mathbb{R}^{|S|}$. As a result, the probabilities for the next state may be determined using the current probabilities $q_i^{(t)}$ and the transition probabilities $p_{i,j}$:

$$\begin{aligned} q_j^{(t+1)} &= \Pr [X_{t+1} = j] \\ &= \sum_{i \in S} \Pr [X_{t+1} = j \mid X_t = i] \times \Pr [X_t = i] \\ &= \sum_{i \in S} p_{i,j} \times q_i^{(t)} \end{aligned} \quad (4)$$

Equivalently this can be expressed using Matrix notation: $q^{(t+1)} = q^{(t)} \times P$.

Distributions π , which do not change after another iteration, are called stationary distributions: $\pi = q^{(t+1)} = q^{(t)}$ or $\pi = \pi \times P$.

If it is feasible to transition from any state to any (other) state, formally $\forall (i, j) \in S^2, \exists n \in \mathbb{N} : p_{i,j}^{(n)} > 0$, the Markov Chain is said to be irreducible. This is true if the state graph of the Markov Chain has a high degree of connectivity.

For irreducible Markov Chains a unique stationary distribution π exists. Another sufficient condition for a unique π is satisfying *detailed balance* as in [1]:

$$\forall i, j \in S : \pi_i \times p_{i,j} = \pi_j \times p_{j,i} \quad (5)$$

A state $i \in S$ is said to be aperiodic if it is feasible to return to a state i to it in any arbitrary number of steps after leaving it, as long as the number is sufficiently big, hence:

$$\exists n_0 \in \mathbb{N} : \forall n \in \mathbb{N}, n \geq n_0 : p_{ii}^{(n)} > 0 \quad (6)$$

For instance, if a state i has a loop around itself, indicating that $p_{i,i} > 0$, it is aperiodic.

If $\forall i \in S, i$ is aperiodic and irreducible, it is said to be ergodic. Regardless of the initial distribution $q^{(0)}$, an ergodic chain always converges to π . Formally:

$$\lim_{t \rightarrow \infty} q^{(t)} = \pi \tag{7}$$

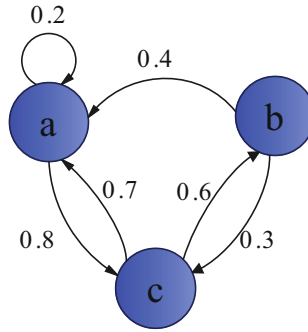


Fig. 2. This is a graphical example of a Markov Chain with three states a, b and c. The edges are annotated with the transition probabilities. The probabilities of the outgoing arrows of each state sum up to 1.

An example of a graphical depiction of an ergodic Markov Chain with three states as nodes and a transition matrix as edges is given in Fig. 2.

$$P = \begin{pmatrix} 0.2 & 0 & 0.8 \\ 0.4 & 0 & 0.6 \\ 0.7 & 0.3 & 0 \end{pmatrix}$$

3 Monte Carlo Simulations

Monte Carlo simulations are probabilistic processes that use repeated random sampling and the law of large numbers to numerically approximate solutions to complex problems. According to [12], given a random variable X , an $\epsilon > 0$ and a $\delta > 0$, the law of large numbers states: If $n \geq \frac{\text{Var}[X]}{\epsilon \delta^2}$ and X_1, \dots, X_n are random variables with the same distribution as X ,

$$\Pr \left[\frac{X_1 + \dots + X_n}{n} - E[X] \geq \delta \right] \leq \epsilon \tag{8}$$

Thus, for any arbitrarily tiny positive precision and error probability, the expected value of X may be computed with a large enough n .

A well-known example is the approximation of the circular number π by sampling n times from a square of uniform distribution with length a . Thus, the number of samples c contained inside a circle with a radius of $\frac{a}{2}$ and centred in the centre of the square is tallied. Pythagoras' theorem may be used to assess

if a sample (x, y) is included inside that circle. With the area of the square $A_{square} = a^2$ and the area of the circle $A_{circle} = \pi \times \left(\frac{a}{2}\right)^2$, the ratio of the areas equals the chance that a sample will be in the circle. Hence,

$$\lim_{n \rightarrow \infty} \frac{c}{n} = E \left[\frac{c}{n} \right] = \frac{\sum_{i=1}^n \text{Pr}[\text{sample } i \text{ in the circle}]}{n} = \frac{A_{circle}}{A_{square}} = \frac{\pi}{4} \quad (9)$$

Thus, for sufficiently large n (at-least 1000 iterations), we have:

$$\pi \approx 4 \times \frac{\text{number of samples in circle}}{\text{total iterations}} =: \hat{\pi} \quad (10)$$

The approximation of π is done by counting samples inside the circle. The result is shown in Fig. 3.

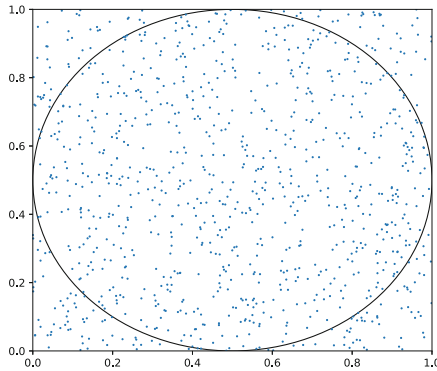


Fig. 3. Approximation of $\hat{\pi}$ by counting samples inside the circle. For $c = 789$ and $n = 1000$, $\hat{\pi} \approx 3.156$.

Monte Carlo techniques may be used with Markov Chains to generate random samples that adhere to a specified probability distribution p^* . The fundamental concept behind so-called Markov Chain Monte Carlo (*MCMC*) approaches is to construct a Markov Chain with a stationary distribution π that closely approximates the desired probability distribution. Following construction, the Markov Chain is executed and the visited states (or a portion of them) are returned as samples. Often, these distributions p^* are complex, and the procedures for building them use basic (proposal) distributions, such as a Gaussian or an unitary distribution [9].

4 Metropolis-Hastings Algorithm

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 challenging. The Metropolis-Hastings algorithm, which serves as the foundation for many MCMC approaches, is one prominent method for creating such a Markov Chain that follows a particular probability distribution. It was first published in 1953 by Metropolis, who used it for computations in Physics [18] while Hastings introduced it in 1970 as an extension to the Metropolis algorithm [11].

4.1 Mathematical Underpinnings

Given any function $p : \mathbb{R}^n \rightarrow \mathbb{R}_0^+$, Metropolis-Hastings delivers samples that follow the distribution specified by that function [19]. Notably, there is no need that p has to be normalised, which means that $\int_{x \in \Omega} p(x) dx = 1$ does not have to hold. The samples continue to conform to the normalised probability distribution function $p^*(x) = \frac{p(x)}{Z}$, where Z is the normalising constant, so that $\int_{x \in \Omega} \frac{p(x)}{Z} dx = 1$. Take note that in this situation, $x = (x_1 \cdots x_n)$ is not a scalar, but a vector in case of multivariate distributions.

The Markov Chain is produced implicitly since no transition matrix is ever computed explicitly and the suggested next state x' and its transition probability are calculated on demand. Given a state $x^{(t)}$ the algorithm offers a subsequent state x' by sampling from a proposal distribution $q(x' | x^{(t)})$. The proposal distribution q must assign a probability greater than zero to states in the target distribution p that has a probability greater than zero. A proposal x' is accepted as next state ($x^{(t+1)} := x'$) with probability:

$$\min \left(1, \frac{p(x') \times q(x^{(t)} | x')}{p(x^{(t)}) \times q(x' | x)} \right) \quad (11)$$

and discarded otherwise, so that the current state will also be the next state ($x^{(t+1)} := x^{(t)}$).

The Metropolis-Hasting pseudo code is described in algorithm 1. The primary distinction between Hastings and the Metropolis method is that whereas Metropolis employed only symmetric proposal distributions, Hastings devised the so-called Hastings adjustment, which allowed for non-symmetric proposal distributions as well.

Algorithm 1. Metropolis-Hastings Method

- 1: Initialize x_0
 - 2: **for** $t = 0, 1, 2, \dots$ **do**
 - 3: $x := x^{(t)}$
 - 4: sample $x' \sim q(x' | x)$
 - 5: acceptance probability $\alpha := \frac{p(x') \cdot q(x|x')}{p(x) \cdot q(x'|x)}$ $r := \min(1, \alpha)$
 - 6: sample $u \sim U(0, 1)$, where U is unitary distribution
 - 7: new sample $x^{(t+1)} := \begin{cases} x' & \text{if } u < r \\ x_t & \text{otherwise} \end{cases}$
 - 8: **end for**
-

It can be proven that the resultant stationary distribution follows $p^*(x)$, since it meets the detailed balancing criterion indicted in equation (5).

This may be readily demonstrated in the situation when $x^{(t)} = i$ and q suggests a $j \neq i$ as in [1]. The proof of the preceding assumption is as follows:

$$\begin{aligned}
 & \Pr [x^{(t)} = i] \times \Pr [x^{(t+1)} = j | x^{(t)} = i] \\
 = & \Pr [x^{(t)} = i] \times \Pr [j \text{ is proposed} | x^{(t)} = i] \\
 & \quad \times \Pr [j \text{ is accepted} | x^{(t)} = i] \\
 = & p^*(i) \times q(j | i) \times \min \left(1, \frac{p(j) \times q(i|j)}{p(i) \times q(j|i)} \right) \\
 = & p^*(i) \times q(j | i) \times \min \left(1, \frac{p^*(j) \times q(i|j)}{p^*(i) \times q(j|i)} \right) \\
 = & \min (p^*(i) \times q(j | i), p^*(j) \times q(i | j)) \\
 = & \min (p^*(j) \times q(i | j), p^*(i) \times q(j | i)) \\
 = & \Pr [x^{(t)} = j] \times \Pr [x^{(t+1)} = i | x^{(t)} = j]
 \end{aligned} \tag{12}$$

Additionally, the constructed Markov Chain is irreducible and ergodic, the resultant distribution is unique, thus convergence to p^* is granted as in [19].

4.2 Optimizations and Challenges

Typically, as with the original Metropolis method, a symmetrical proposal distribution q is selected, of $q(x' | x) = q(x | x')$ form. Frequently, a Gaussian distribution with constant or adaptive variance is utilised, centred on x so that:

$$q(x' | x) \sim \mathcal{N}(x' | x, \Sigma) \tag{13}$$

The covariance matrix Σ must be optimized to produce acceptable acceptance rates that are neither too low (which results to a lot of duplicates) nor too high where the state space is explored very slowly). Murphy suggests aiming for acceptance rates of between 25% and 40% in [19].

However, the algorithm is not perfect or fine-tuned. Rather of being independent, as needed for samples, the states are strongly connected, or auto-correlated. Depending on the proposal function, states close i are more likely to be sampled than others. One solution is to return just a sample after every n -th step, which

is termed thinning. This does not totally cure the issue, but it does reduce the association. Another issue is that the first samples heavily rely on the starting condition (or distribution). There is a “burn-in” phase when no samples are created for the first k steps. Forgetting the starting state assures the distribution of the Markov Chain is close to the true distribution p^* . For tiny Markov Chains that meet certain criteria of proximity to the true distribution some k can be computed. For larger Markov Chains, heuristics are used instead as in [14]. A trace plot may be made by running many chains. If the plots overlap and converge, the chain has mingled.

Tuning these parameters n and k is difficult, as there are trade-offs: The higher they are, the more steps are not considered, so it takes more steps and therefore time to create samples, which reduces computational efficiency. The lower they are, the more correlated and therefore of worse quality these samples are. To get sufficient samples, run numerous chains and sample their states. Murphy offers three 100.000-step chains, with half discarded and the remainder sampled. The more dimensions x has, the more likely suggested samples will be rejected. Bishop suggests picking the Gaussian scale based on the least standard deviation of each dimension, as seen in Fig. 4.

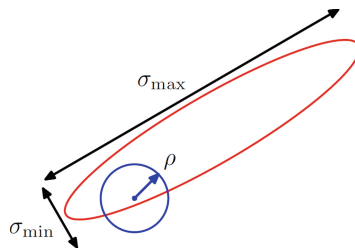


Fig. 4. A MH two-dimension function (red ellipse). A Gaussian with standard deviation $\rho \sim \sigma_{\min}$ is used as proposal distribution to avoid high rejection rates (blue circle) [1]. (Color figure online)

4.3 Gibbs Sampling

Gibbs Sampling, as explained in algorithm 2, is a common specific instance of MH that should not be overshadowed. The notion is that at each step, one (or a small subset) of the components i is updated by sampling from everything except i and replacing it with the most current values. Thus, to update the first component of $x^{(t)}$, $x_1^{(t+1)}$ is sampled from $p(x_1^{(t+1)} | x_2^{(t)}, \dots, x_n^{(t)})$. This is essentially MH, where every proposal is approved since $\alpha = 1$ is always true [19]. This simplifies the collection of samples, but their auto-correlation is increased.

Algorithm 2. Gibbs Sampling Method

```

1: Initialize  $x_0$ 
2: for  $t = 0, 1, 2, \dots$  do
3:   sample:  $x_1^{(t+1)} \sim p\left(x_1^{(t+1)} \mid x_2^{(t)}, \dots, x_n^{(t)}\right)$ 
4:    $x_2^{(t+1)} \sim p\left(x_2^{(t+1)} \mid x_1^{(t+1)}, x_3^{(t)}, \dots, x_n^{(t)}\right) \vdots$ 
5:    $x_j^{(t+1)} \sim p\left(x_j^{(t+1)} \mid x_1^{(t+1)}, \dots, x_{j-1}^{(t+1)}, x_{j+1}^{(t)}, \dots, x_n^{(t)}\right) \vdots$ 
6:    $x_n^{(t+1)} \sim p\left(x_n^{(t+1)} \mid x_1^{(t+1)}, \dots, x_{n-1}^{(t+1)}\right)$ 
7: end for

```

5 Applications of MCMC Methods

MCMC methods have big advantages over other sampling methods like rejection sampling, as they scale well with higher dimensions. Therefore, sampling with MH has many applications, several of which are presented in this paper. In this section the three major applications of MCMC methods we focus on are: integral estimation, simulated tempering and text decryption.

5.1 Estimation of Integrals

Particularly in Physics, where the Metropolis method originated, many integrals must be approximated, often with unknown normalisation factors for marginal likelihood calculations in order to detect gravitational waves [10]. A multidimensional integral may be estimated using Metropolis-Hastings [9]. For example, estimating the value of the following integral:

$$s := \int p(x) \times f(x) dx = E_p[f(x)] \tag{14}$$

with p being a normalized probability distribution. Then after using MH to draw samples $x^{(1)}, \dots, x^{(n)}$ from p , s can be estimated by

$$\hat{s} = \frac{1}{n} \sum_{i=1}^n f\left(x^{(i)}\right) \tag{15}$$

One can show that the expectation value is the same:

$$E_p[\hat{s}] = E_p \left[\frac{1}{n} \sum_{i=1}^n f\left(x^{(i)}\right) \right] = \frac{1}{n} \sum_{i=1}^n E_p \left[f\left(x^{(i)}\right) \right] = \frac{1}{n} \sum_{i=1}^n s = s \tag{16}$$

So given only a function $g(x)$ and seeking an estimate for $s_g := \int g(x) dx$, a factorization $g(x) = f(x) \times p(x)$ with $p(x)$ being a valid probability distribution has to be found. Alternatively, one can sample from any probability distribution function $h(x)$ and use a technique called importance sampling.

As $g(x) = h(x) \times \frac{g(x)}{h(x)}$ the estimator can be modified:

$$\hat{s}_g = \frac{1}{n} \sum_{i=1}^n \frac{g(x^{(i)})}{h(x^{(i)})} \tag{17}$$

Again, $E_h[\hat{s}_g] = s_g$ holds. Ideally, $h(x)$ is (up to a scaling factor) similar to $g(x)$ [9]. This proves to be powerful method for approximating any difficult integral. To illustrate it, an integral of the two dimensional function

$$g(x_1, x_2) = e^{-x_1^2 - x_2^2} \tag{18}$$

will be estimated:

$$s := \int_{x_1 \in (0,1), x_2 \in (0,1)} g(x_1, x_2) dx_1 dx_2 \tag{19}$$

For importance sampling, we set:

$$h(x_1, x_2) = (2 - x_1 - x_2) \tag{20}$$

Note that: $\int_{x_1 \in (0,1), x_2 \in (0,1)} h(x_1, x_2) dx_1 dx_2 = 1$ and $h(x_1, x_2) > 0$ for the given integral. Thus h is a valid distribution function. Now, MH with a proposal function $q(x' | x) \sim \mathcal{N}(x' | x, 0.2)$ is used to draw 200 to 1000 samples from h . The algorithm is initialized by sampling from a normal distribution centred at $(0, 0)$. This achieves an acceptance rate of 35.1% for $n = 200$ samples and 34.17% for $n = 1000$ samples. Using the samples and equation (17), the approximation results in $\hat{s}_h \approx 0.569$. This is relatively accurate, as the exact value is $s \approx 0.5677$. A visualization can be found in Figs. 5 and 6. The z-value of a sample x is $h(x)$. The density of the samples is higher in regions with high h -values and low in the others, as expected.

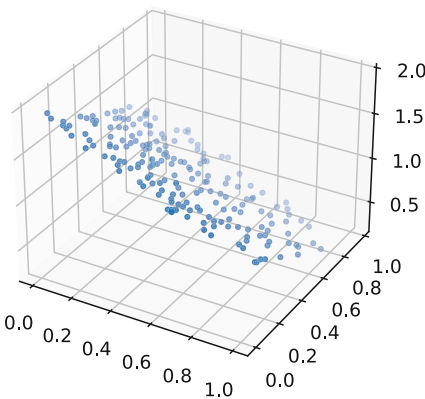


Fig. 5. MH on 200 samples.

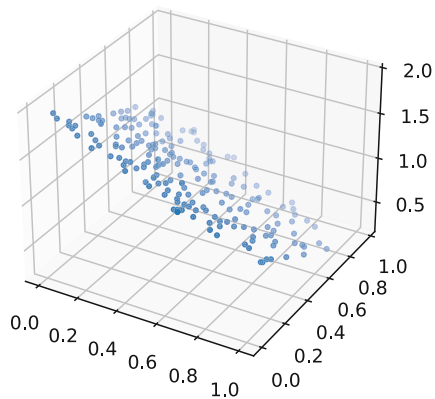


Fig. 6. MH on 1000 samples.

5.2 Simulated Tempering

However, MH may be utilised to solve optimization issues involving non-convex functions when a global minimum must be established and conventional gradient descent algorithms fail because they get caught in one of the several local minima [19]. To discover the minimum, one would use a distribution that assigns a high probability to f values that are small. A popular option is the Boltzmann distribution.

$$p(x) = e^{-\frac{f(x)}{T}} \tag{21}$$

Utilizing the Boltzmann distribution here the requirement is to have an adjustable temperature T . This T value is lowered over time as Metropolis-Hastings executes the chain using a process called simulated annealing. This results in decreasing uphill movements over time, increasing the value of the function. The states converge to the global minimum as the probability of the chain being in the region with the greatest probabilities increases, without being trapped in a local minimum. Thus, analogous to physics, although huge motions are initially permitted to explore the state space, the system “cools down” and converges. At the conclusion of the random walk, the best x observation is returned. Once again, the starting temperature and the manner in which T is lowered are tuneable. Figure 7 shows a representation of various temperatures.

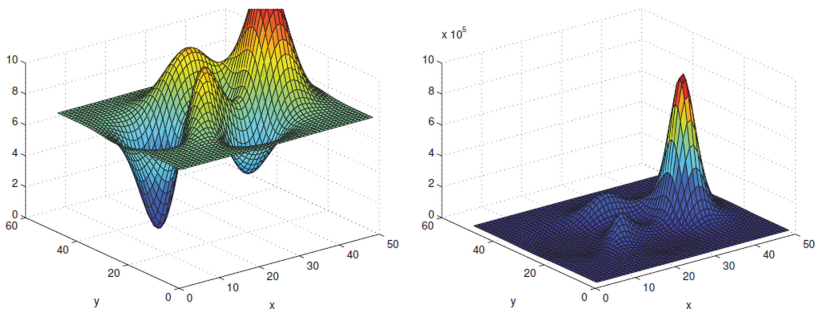


Fig. 7. Plot of the Boltzmann function for a function at a high temperature, where many areas have high values (left) and a low temperature, where only areas close to the optimal receive high values (right) [19].

Simulated tempering techniques, such as MH, may be used not just to continuous functions in \mathbb{R}^n , but also to discrete functions. This knowledge may be used to provide estimates for a given Traveling Salesman Problem (TSP) optimization [16]. TSP searches for the shortest route across a graph that visits each node precisely once. As a result, the states of this issue are represented by pathways across the graph that pass through each node precisely once. The length of these pathways added into the Boltzmann function is the function supplied to MH, which is no longer a legitimate probability distribution. Proposals are

formed by rearranging the order in which two (or more) nodes in a given route are visited. This heuristic produces reasonable approximations, despite the fact that there is no optimum solution to the otherwise NP-hard issue.

5.3 Text Decryption

Another notable example of MH is interpreting documents, such as jail prisoners' secret code [6]. Given a text that has been ciphered by substituting other symbols Y for the letters S in the underlying text, it is feasible to learn the inverse f of the cypher function $c : S \rightarrow Y$ with remarkable accuracy.

By analysing the probabilities of each character y following another x in commonly used English texts, one may establish a decent estimate of a decode plausibility of a decipher function. This pre-supposes that the provided text is comparable to those, i.e. that the same language and proficiency level are employed. As a result, a transition matrix can be learned:

$$m_{x,y} = \Pr[\text{next character is } y \mid \text{current character is } x] \quad (22)$$

Using this matrix M , a measure of plausibility for a decipher function f can be defined as:

$$Pl(f) = \prod_i m_{f(s_i), f(s_{i+1})} \quad (23)$$

Pl is entered into Algorithm 1 as the probability function. The proposal f' is created from state f by randomly swapping two assignments in f . As a result, the proposal distribution is symmetrical, and the acceptance probability decreases to:

$$\min\left(1, \frac{Pl(f')}{Pl(f)}\right) \quad (24)$$

However, there is a difference here: In this MH-walk, the states are possible decipher functions f , so the states are not in \mathbb{R}^n , but in the space of all bijective functions $\{f : Y \rightarrow S\}$. The first f simply allocates a distinct character in S to each symbol in Y . Within 2000 steps of this run, a sufficiently good deciphering function f was discovered, where the assignments are no longer often changing, indicating that the Markov Chain had converged. This is a surprising achievement, given there are around 40 potential functions in the search space for the approximately 40 distinct characters found in typical texts (letters, numerals, spaces, punctuation characters, etc.).

6 Conclusion

MCMC techniques seem to be quite beneficial in a wide variety of applications. The Metropolis-Hastings algorithm is among the top of the list of great algorithms of 20th century scientific computing [5], and its versions are critical for Bayesian statistics and machine learning. Nonetheless, MCMC approaches are approximate, and as a consequence of unpredictability, variations from the

proper conclusions are possible. As a result, MCMC should be used sparingly and only in the absence of better alternatives, as no assurances can be made. Thus, assuming computing feasibility, this should be favoured over MCMC for integrals that can be solved analytically. For practical applications, more sophisticated variations of Metropolis-Hastings are employed as indicated in [13], since they need fewer steps and hence provide better samples, while also being tuned to avoid numerical difficulties, which are not discussed here. Alternatively, performance may be optimised by dynamically adjusting parameters, particularly the covariance matrix, without switching the distribution as the parameters vary over time. Additionally, various changes to Metropolis-Hastings are required for low correlations in higher dimensions.

References

1. Bishop, C.M.: Pattern Recognition and Machine Learning (Information Science and Statistics). Springer-Verlag, Berlin, Heidelberg (2006)
2. Brémaud, P.: Markov Chains: Gibbs Fields, Monte Carlo Simulation, and Queues, vol. 31. Springer, New York (2013). <https://doi.org/10.1007/978-3-030-45982-6>
3. Chen, J., Rosenthal, J.S.: Decrypting classical cipher text using Markov chain Monte Carlo. *Stat. Comput.* **22**(2), 397–413 (2012). <https://doi.org/10.1007/s11222-011-9232-5>
4. Chib, S., Greenberg, E.: Understanding the metropolis-hastings algorithm. *Am. Stat.* **49**(4), 327–335 (1995). <https://doi.org/10.1080/00031305.1995.10476177>
5. Cipra, B.A.: The best of the 20th century: editors name top 10 algorithms. *SIAM News* **33**(4), 1–2 (2000)
6. Diaconis, P.: The Markov chain Monte Carlo revolution. *Bull. Am. Math. Soc.* **46**(2), 179–205 (2009)
7. Gelfand, A.E.: Gibbs sampling. *J. Am. Stat. Assoc.* **95**(452), 1300–1304 (2000). <https://doi.org/10.1080/01621459.2000.10474335>
8. Geman, S., Geman, D.: Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Trans. Pattern Anal. Mach. Intell. PAMI* **6**(6), 721–741 (1984). <https://doi.org/10.1109/TPAMI.1984.4767596>
9. Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning. MIT press, Cambridge (2016)
10. Haasteren, R.V.: Marginal likelihood calculation with MCMC methods. In: Gravitational Wave Detection and Data Analysis for Pulsar Timing Arrays, pp. 99–120. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-642-39599-4_5
11. Hastings, W.K.: Monte Carlo sampling methods using Markov chains and their applications. *Biometrika* **57**, 97–109 (1970)
12. Huang, H., Yang, W.: Strong law of large numbers for Markov chains indexed by an infinite tree with uniformly bounded degree. *Sci. China Ser. A: Math.* **51**(2), 195–202 (2008)
13. Kaji, T., Ročková, V.: Metropolis-hastings via classification. *J. Am. Stat. Assoc.*, 1–33 (2022). <https://doi.org/10.1080/01621459.2022.2060836>
14. Karras, C., Karras, A.: DBSOP: an efficient heuristic for speedy MCMC sampling on polytopes. arXiv preprint [arXiv:2203.10916](https://arxiv.org/abs/2203.10916) (2022). <https://doi.org/10.48550/arXiv.2203.10916>

15. Karras, C., Karras, A., Sioutas, S.: Pattern recognition and event detection on IoT data-streams. arXiv preprint [arXiv:2203.01114](https://arxiv.org/abs/2203.01114) (2022). <https://doi.org/10.48550/arXiv.2203.01114>
16. Kirkpatrick, S., Gelatt, C.D., Jr., Vecchi, M.P.: Optimization by simulated annealing. *Science* **220**(4598), 671–680 (1983)
17. Martino, L., Elvira, V., Luengo, D., Corander, J., Louzada, F.: Orthogonal parallel MCMC methods for sampling and optimization. *Digital Signal Process.* **58**, 64–84 (2016). <https://doi.org/10.1016/j.dsp.2016.07.013>
18. Metropolis, N., Rosenbluth, A.W., Rosenbluth, M.N., Teller, A.H., Teller, E.: Equation of state calculations by fast computing machines. *J. Chem. Phys.* **21**(6), 1087–1092 (1953)
19. Murphy, K.P.: *Machine Learning: A Probabilistic Perspective*. MIT press, Cambridge (2012)
20. Revuz, D.: *Markov Chains*. Elsevier, Amsterdam (2008)
21. Ripley, B.D.: *Stochastic Simulation*. John Wiley & Sons, Hoboken (2009)
22. Wolfinger, R., O’connell, M.: Generalized linear mixed models a pseudo-likelihood approach. *J. Stat. Comput. Simul.* **48**(3–4), 233–243 (1993). <https://doi.org/10.1080/00949659308811554>
23. Xu, J.-G., Zhao, Y., Chen, J., Han, C.: A structure learning algorithm for bayesian network using prior knowledge. *J. Comput. Sci. Technol.* **30**(4), 713–724 (2015). <https://doi.org/10.1007/s11390-015-1556-8>