MARKOV CHAIN MONTE CARLO

GRACE CAI

Abstract. Markov chain Monte Carlo is a modern method used to find information about a distribution with random sampling. We will explore the theory behind MCMC and one of its most popular algorithms, the Metropolis-Hastings algorithm, as well as look at a couple of simple examples and real-life applications.

1. Introduction

Markov chain Monte Carlo (MCMC) is a computer-driven sampling method under the Monte Carlo method that utilizes Markov chains. The Monte Carlo method came soon after the invention of computers and it simulates complex models to approximate numerical information about distributions by generating random samples from a distribution and using numerical simulation repeatedly. The MCMC method was invented by Stanislaw Ulam while working on nuclear weapons at the Los Alamos National Laboratory. His idea for this method came from playing solitaire and trying to find a combinatorial way to calculate the probability of winning, but ultimately being unable to and using simulations to model solitaire instead. The name Monte Carlo comes from the Monte Carlo Casino in Monaco. It is used for situations that demonstrate the Markov property so a Markov chain can be created to model it. This means that each new sample taken is dependent only on the previous sample. When using the MCMC method, not all mathematical properties of the distribution must be known since it is taking random samples from the distribution. This makes MCMC a powerful tool in many real-world situations where there is enough data for many random samples to be drawn from and when the distribution's equations can only be easily handled with a computer.

2. Markov Chain Background for MCMC

In order to understand MCMC, we need to gain the necessary knowledge about Markov chains and the Monte Carlo method separately. We will begin by defining Markov chains and their properties.

Definition 2.1. A sequence $X_1, X_2, \ldots$ of random elements in some set Ω (a stochastic process) is a *Markov chain* if the conditional distribution of X_{n+1} given $X_1, \ldots, X_n$ depends on X_n only. The set in which the random elements take values is called the *state space* of the Markov chain.

A probability distribution is a function that gives us the probability of different possible outcomes of a certain event. Both the initial distribution—the marginal distribution of X_i—and the conditional distribution of X_{n+1} given X_n determine the joint distribution of a Markov chain.

Date: November 23, 2020.

Definition 2.2. A Markov chain with transition matrix P is *stationary* if there exists a *stationary distribution* π such that

$$\pi = \pi P.$$

This property is very important in MCMC because it samples the stationary distribution regardless of the stationarity of the Markov chain. This property is also present in the Markov chains used in the ordinary Monte Carlo method which we will describe in the next section. We will also be working with Markov chains where the probability of transitioning from state i to state j is equal to the probability of transitioning from state j to state i for any two states i, j.

Definition 2.3. A Markov chain $X_1, X_2, \ldots$ with transition matrix P and stationary distribution π is *reversible* if and only if, for all states $i, j \in \Omega$, we have $\pi_i p_{ij} = \pi_j p_{ji}$. This also means that the distribution of any pairs (X_i, X_{i+1}) is exchangeable.

This property is used in the Metropolis-Hastings algorithm, a widely-used MCMC method, which we will define later. Another property used in the Metropolis-Hastings algorithm is ergodicity.

Definition 2.4. A Markov chain is said to be *ergodic* if it is *aperiodic*— meaning the process does not return to the same state i at fixed intervals—and *irreducible* —it is possible to reach any state i from some state j in some integer t number of steps so we have $p_{ij}^{(t)} > 0$.

An important theorem involved in the MCMC method (and probability theory in general) is the law of large numbers (LLN). This theorem has a weak form and a strong form, but we will only be using the strong form in this article:

Theorem 2.1. *(The Strong Law of Large Numbers) Let $X_1, \ldots, X_n$ be n identically and independently distributed (iid) random variables such that each have defined and finite $\mathbb{E}[X_i] = \mu$. We call $S_n = \sum_{i=1}^{n} X_i$ the sample sum and $Y_n = \frac{S_n}{n}$ the sample average. Then, Y_n almost surely converges to μ. In mathematical terms, we have*

$$\mathbb{P}(\omega \in \Omega : \lim_{n\to\infty}[Y_n(\omega)] = \mu) = 1,$$

$$Y_n \xrightarrow{a.s.} \mu, n \to \infty.$$

The "a.s." over the arrow means almost surely. This means that as our sample size grows larger and larger, the sample average gets closer to the expected value. This theorem is the reasoning behind using more samples in an MCMC algorithm. We will not prove this theorem in this article. A proof can be found in [Rot20].

3. The Monte Carlo Method

Now, let's look at the Monte Carlo method without the Markov chain aspect, sometimes called ordinary Monte Carlo (OMC). Ordinary Monte Carlo is a special case of MCMC where the random elements $X_1, X_2, \ldots$ are identically and independently distributed (iid). In this case, the Markov chain modeling the situation must be stationary and reversible. The Monte Carlo method can be used to sample from, or to estimate an expected value with respect to, a high dimensional probability distribution. The method is generally described in [Has70] as follows:

- To simplify the distribution, factorize it into the product of one-dimensional conditional distributions and take samples from there.

- Use importance sampling to possibly reduce variation. In other words, to evaluate the integral

$$J = \int f(x)p(x)dx = \mathbb{E}_p(f),$$

where $p(x)$ is a probability density function (PDF), we obtain samples from a distribution $q(x)$ and use the estimate $\hat{J}_2 = \sum\{\frac{f(x_i)p(x_i)}{q(x_i)N}$ instead of taking independent samples $x_1, \ldots, x_N$ from $p(x)$ and using the estimate $\hat{J}_1 = \sum\{\frac{f(x_i)}{N}$. Doing this helps when sampling from $q(x)$ is easier than sampling from $p(x)$. However, the weights $w(x_i) = \frac{p(x_i)}{q(x_i)}$ for reasonable values of N may all be either extremely small, or a few could be extremely large. When we estimate the probability of an event A, this issue has little effect because we only want the values $w(x)$ such that $x \in A$.
- Use a simulation technique; if sampling directly from $p(x)$ is difficult or $p(x)$ is unknown, sample from some distribution $q(y)$ and get the corresponding x values as a function of the y values. If we want samples from the conditional distribution of $x = g(y)$, given $h(y) = h_0$, then the simulation technique will not be satisfactory if $\mathbb{P}h(y) = h_0$ is small, since the condition $h(y) = h_0$ will be rarely if ever satisfied even when the sample size from $q(y)$ is large.

Example 3.1. A simple application of the Monte Carlo method is estimating π through many random samplings. We can imagine a 1×1 square with a quarter circle with radius 1 aligned with the left and bottom sides of the square as shown in figure (1) and take our random samples from any point (x, y) in the square where $0 \leq x, y \leq 1$. From the area of the quarter circle and the square, we have the ratio $\frac{1}{4}\pi : 1$ or $\pi : 4$ so if we have a large sample size, the ratio of the number of points inside the quarter circle to the points in the square should also have a $\pi : 4$ ratio. To tell whether the random point (x, y) is in the quarter circle, we see if it satisfies the inequality $x^2 + y^2 \leq 1$. We estimate π by multiplying this ratio by 4.

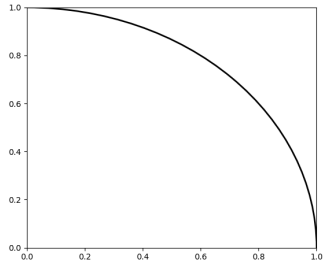

Figure 1. Unit square with quarter of unit circle inscribed.

This is the psuedocode for our program which can be run using Python and graphed with the Python library MatPlotLib:

(1) Generate random point (x, y), $0 \leq x, y \leq 1$ and plot on graph
(2) If point is in quarter circle, $x^2 + y^2 \leq 1$, add 1 to running total
(3) Repeat for sample size n
(4) Divide total by n and multiply by 4
(5) Result should be very close to π depending on n

Running this code with a sample size of 10,000 gives us a π estimation of 3.15052 and the graph (2):

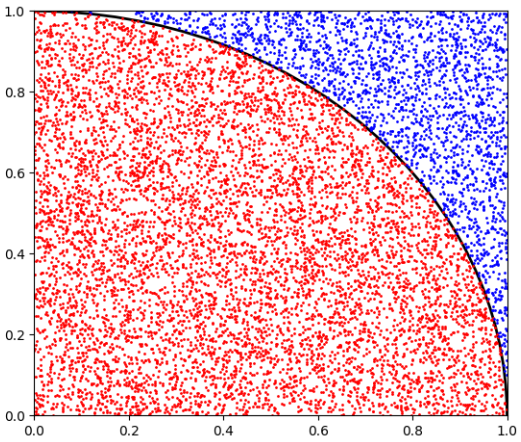

Figure 2. Graph of 10,000 random points to estimate π.

4. Markov Chain Monte Carlo

Now that we have the necessary background, we will introduce Markov Chain Monte Carlo and look at one of its most useful algorithms, the Metropolis-Hastings (M-H) algorithm. This method uses the Markov chain central limit theorem (CLT) which is similar to the classical central limit theorem except for the fact that the sequence does not need to be independent and it involves a real-valued function. This theorem describes the convergence of Markov chains used in MCMC methods.

Theorem 4.1. *(The Markov Chain Central Limit Theorem) If the sequence $X_1, X_2, \ldots$ of random elements of a Markov chain satisfies the following conditions:*

- *it has a stationary distribution,*
- *the distribution of X_1 is the stationary distribution so the sequence is identically distributed,*
- *and we have some real-valued function g such that $var(g(X_1)) < \infty$,*

and we let

- $\mu = \mathbb{E}(g(X_1))$,
- $\sigma^2 = var(g(X_1)) + 2\sum_{k=1}^{\infty} cov(g(X_1), g(X_{k+1}))$,
- $\hat{\mu_n} = \frac{1}{n}\sum_{k=1}^{n} g(X_k)$,

then as n approaches infinity, we have

$$\hat{\mu_n} \approx Normal\left(\mu, \frac{\sigma^2}{n}\right)$$

or

$$\sqrt{n}(\hat{\mu_n} - \mu) \xrightarrow{D} Normal\left(0, \frac{\sigma^2}{n}\right)$$

where $\xrightarrow{D}$ *denotes convergence in distribution.*

We will not prove this theorem (a proof can be found in [Kur81]) but we can demonstrate this with the Metropolis-Hastings algorithm in the next section.

5. The Metropolis-Hastings Algorithm

One popular algorithm that uses the MCMC method is the Metropolis-Hastings (M-H) algorithm. The goal of this algorithm is to create a Markov chain based on a desired distribution $P(x)$. It works by running a Markov process which tends towards a unique stationary distribution $\pi(x) = P(x)$. A unique stationary distribution $\pi(x)$ exists when the transition $x \to y$ is reversible and the Markov chain is ergodic. Therefore, we only consider reversible and ergodic Markov chains for simplicity.

The M-H algorithm includes two steps: a proposal and an acceptance-rejection. The proposal distribution $q(y \mid x)$ is the conditional probability of proposing a state y given x, and the acceptance distribution $\alpha(x, y)$ is the probability to accept the proposed state y. Then, the algorithm works by initializing a starting point x_0 and repeating the following steps, each iteration being a M-H update, for $n = 1, 2, ..., N$:

(1) **Proposal**: Given state x_n, generate random proposal state y according to $q(x_n|y)$, the probability distribution.
(2) Calculate the acceptance probability of moving from x_n to y

$$\alpha(x_n n, y) = \min\left\{1, \frac{\pi(y)q(y, x_n)}{\pi(x_n)q(x_n, y)}\right\}$$

and $\alpha = 1$ if $\pi(x_n)q(x_n, y) = 0$. The fraction $\frac{\pi(y)q(y,x_n)}{\pi(x_n)q(x_n,y)}$ is called the *Hastings ratio.*
(3) **Acceptance-rejection**: Generate a random u from the uniform distribution on [0,1], $U[0, 1]$. If $u \leq \alpha(x_n n, y)$, accept the new state so $x_{n+1} = y$. Otherwise, reject y and set $x_{n+1} = x_n$.
(4) Return the values $\{x_1, x_2, ..., x_N\}$

In other words, the M-H algorithm gets a proposal value y to jump to and either accepts or rejects that value depending on how high the probability of moving to y is. The values $\{x_1, x_2, ..., x_N\}$ return from the algorithm form a Markov chain whose empirical distribution will approach $P(x)$. An empirical distribution is a function used to describe a sample of observations of a given variable. The number of steps N required to effectively estimate $P(x)$ depends on the relationship between $P(x)$ and the proposal distribution and the desired accuracy of estimation. Including more steps N results in a more accurate distribution of the sample and lessens the effects of the starting position. This is a demonstration of the SLLN. Now, we will go into the conditions of convergence for the M-H algorithm based on the properties of certain distributions. We will prove two theorems in this area that result from the M-H algorithm. First, we will state and prove the Metropolis-Hastings theorem

which says that a Markov chain produced using the M-H Algorithm is reversible with respect to a distribution having an unnormalized density [BGJM11, section 1.12.2].

Theorem 5.1. *(The Metropolis-Hastings Theorem) The Metropolis-Hastings update is reversible with respect to h, that is, the transition probability that describes the update is reversible with respect to the distribution having unnormalized density h.*

Proof. Let x_n be the current state and y_n the proposal state so we have $x_n = x_{n+1}$ whenever the proposal is rejected. Thus, the distribution of (x_n, x_{n+1}) given rejection is exchangeable and the Markov chain is reversible. Then, we must show that (x_n, y_n) is exchangeable given acceptance so

$$\mathbb{E}\{f(x_n, y_n)a(x_n, y_n)\} = \mathbb{E}\{f(y_n, x_n)a(x_n, y_n)\} \tag{1}$$

for any function f that has expectation (we assume x_n has the desired stationary distribution). That is, we must show we can interchange arguments of f in

$$\int\int f(x, y)\pi(x)a(x, y)q(x, y)dxdy \tag{2}$$

(with integrals replaced by sums if the state is discrete), and that follows if we can interchange x and y in

$$h(x)a(x, y)q(x, y) \tag{3}$$

because we can exchange x and y in (2), x and y being random variables. Clearly only the set of x and y such that $h(x) > 0$ and $q(x, y) > 0$ and $a(x, y) > 0$ contributes to the integral or (in the discrete case) sum (2), and these inequalities further imply $h(y) > 0$ and $q(y, x) > 0$. Thus we may assume these inequalities, in which case we have

$$r(y, x) = \frac{1}{r(x, y)}$$

for all such x and y. Suppose $r(x, y) \leq 1$, so $r(x, y) = a(x, y)$ and $a(y, x) = 1$. Then

$$\begin{aligned} h(x)a(x, y)q(x, y) &= h(x)r(x, y)q(x, y) \\ &= h(y)q(y, x) \\ &= h(y)q(y, x)a(y, x) \end{aligned}$$

Conversely, suppose $r(x, y) > 1$, so $a(x, y) = 1$ and $a(y, x) = r(y, x)$. Then

$$\begin{aligned} h(x)a(x, y)q(x, y) &= h(x)q(x, y) \\ &= h(y)r(y, x)q(y, x) \\ &= h(y)a(y, x)q(y, x) \end{aligned}$$

In both cases we can exchange x and y in (3) and the proof is done. □

Depending on the properties of the Markov kernel—a map similar to a transition matrix used for finite state spaces—$q(x, y)$, we can also determine two other convergence properties of the M-H algorithm. The conditions for convergence of the M-H algorithm that we will prove are explained in [RS94].

Let $q : D \times D \to \mathbb{R}^+$, $\mathbb{R}^+$ being the positive real numbers, be a Markov chain kernel(with respect to measure V) with $D = \{x \in \mathbb{R}^+; \pi(x) > 0\}$. With $\alpha : D \times D \to [0, 1]$ as described

in the steps of the M-H algorithm. We also define $K_H : D \times D \to \mathbb{R}^+$ by $K_H(x, y) = q(x, y)\alpha(x, y)$. This is a (substochastic) kernel governing moves of the chain $X^0, X^1, \cdot$ from x to y which are 'accepted' according to the probability $\alpha(x, y)$ as previously explained in the steps of the M-H algorithm. It is straightforward to check that π is an invariant distribution of the chain defined by K_H. For general measures V, the convergence properties of the M-H algorithm are inherited from the properties of q as follows.

Theorem 5.2.

i If q is aperiodic; or $P(X^t = X^{t-1}) > 0$, for some $t \geq 1$ then the Metropolis- Hastings algorithm is aperiodic.

ii If q is π-irreducible and $q(x, y) = 0$ if and only if $q(y, x) = 0$, then the Metropolis-Hastings algorithm is π-irreducible.

Proof.

i If the M-H chain is periodic then we cannot have $P(X^t = X^{t-1}) > 0$ for any $t \geq 1$ because the chain can only come back to a certain state after its period $k > 1$. Therefore the chain is aperiodic because of q.

ii The condition $q(x, y) = 0$ if and only if $q(y, x) = 0$ implies that $\alpha(x, y) > 0$ for all $x, y \in D$. Let $K_H^{(t)}, q^{(t)}$ be the iterated kernels obtained by setting K equal to K_H and q, respectively and define $U_x^{(t)} = \{y, K_H^{(t)}(x, y) > 0\}, V_x^{(t)} = \{y; q^{(t)}(x, y) > 0\}$. We show by induction that $U_x^{(t)} \supseteq V_x^{(t)}$ for all $t \geq 1$. Suppose, therefore, that $U_x^{(t)} \supseteq V_x^{(t)}$ and consider $z \in V_x^{(t+1)}$, which implies that

$$\int q^{(t)}(x, y)q(y, z)dv(y) > 0$$

However, if $z \notin U_x^{(t+1)}$ the support of the function

$$q^{(t)}(x, \cdot)q(\cdot, z)\alpha(\cdot, z)$$

has v-measure 0, which implies that the support of the function

$$K_H^{(t)}(x, \cdot)q(\cdot, z),$$

also has v-measure 0, contradicting the above inequality. Since $U_x^{(t)} \supseteq V_x^{(t)}$, the result follows that the M-H algorithm is π-irreducible.

□

Now, let's demonstrate the Markov chain CLT from the previous section.

Example 5.1. We model coin tosses with the following psuedocode that generates a probability density function (PDF) close to the normal distribution depending on the sample size:

(1) Generate random starting point based on given parameters $\mu = 0$ and $\sigma = 1$.
(2) Add current point to state space
(3) Generate probabilities of staying in current state or moving to next state from normal distribution
(4) Use probabilities to find probability of accepting movement p
(5) Generate random probability $\mu \in (0, 1)$
(6) If $\mu \geq p$ then return false for tails
(7) Otherwise, return true for heads
(8) Repeat steps 2-7 for sample size N
(9) Return entire state space to graph distribution
(10) Graph normal distribution curve

This code draws the following graph (3) using MatPlotLib with 20 "bins" to group our data (increasing the number of bins will make the histogram closer to the normal distribution graph):

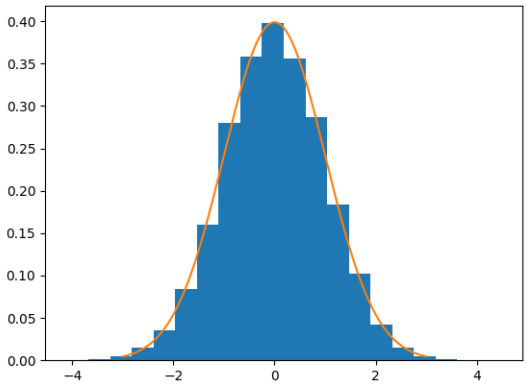

Figure 3. The distribution of the coin tosses (blue) compared to the normal distribution (orange).

The Metropolis-Hastings algorithm is a general MCMC methods, but there are others such as the Metropolis algorithm and the Gibbs sampler both of which are special cases of the M-H algorithm.

6. Modern Applications of MCMC

MCMC algorithms are often applied to solve integration and optimization problems in large dimensional spaces. These two types of problem play a fundamental role in machine learning, physics, statistics, econometrics and decision analysis. A substantial result found in the study of MCMC comes from a paper which describes a general method, suitable for fast electronic computing machines, of calculating the properties of any substance which may be considered as composed of interacting individual molecules [MRR$^+$53]. The method described in this paper consists of a modified Monte Carlo integration over configuration space.

References

[BGJM11] Steve Brooks, Andrew Gelman, Galin Jones, and Xiao-Li Meng. *Handbook of markov chain monte carlo*. CRC press, 2011.

[Has70] W Keith Hastings. Monte carlo sampling methods using markov chains and their applications. 1970.

[Kur81] Thomas G Kurtz. The central limit theorem for markov chains. *The Annals of Probability*, pages 557–560, 1981.

[MRR+53] Nicholas Metropolis, Arianna W Rosenbluth, Marshall N Rosenbluth, Augusta H Teller, and Edward Teller. Equation of state calculations by fast computing machines. *The journal of chemical physics*, 21(6):1087–1092, 1953.

[Rot20] Andrew Rothman. Proof of the law of large numbers part 2: The strong law, Jun 2020. URL: https://towardsdatascience.com/proof-of-the-law-of-large-numbers-part-2-the-strong-law-356aa608ca5d.

[RS94] G.O. Roberts and A.F.M. Smith. Simple conditions for the convergence of the gibbs sampler and metropolis-hastings algorithms. *Stochastic Processes and their Applications*, 49(2):207 – 216, 1994. URL: http://www.sciencedirect.com/science/article/pii/0304414994901341, doi:https://doi.org/10.1016/0304-4149(94)90134-1.