Monte Carlo Methods & Exploration of the Metropolis-Hastings Algorithm

Hisham Bhatti

May 20, 2023

Abstract

Scientists are often presented with a complicated domain and function where standard closed-form integration techniques fail. Monte Carlo Integration aims to overcome this challenge by using random sampling to provide a probabilistic approximation to a deterministic problem. With this method, we then formally prove that the average error is 0, with the average magnitude decreases as a function of $1/\sqrt{N}$ of sample size N. Applying this algorithm allows us to compute the volume of the 10-dimensional unit sphere to within 0.2% of the actual value. We then re-frame this problem in the context of generating random, representative samples from an unknown distribution. The Metropolis-Hastings algorithm provides one such approach, constructing a Markov Chain with each sample conditioned on its prior. Its usage of correlated samples creates a distribution that converges much quicker to the unknown distribution than standard rejection-sampling. We conclude with a comparison between rejection sampling and the Metropolis-Hastings Algorithm in the context of computing an expected value of an unknown distribution, highlighting the increased accuracy and stronger rate of convergence of this method.

1 Motivation

Suppose you were asked to compute the volume of the 2-dimensional unit ball. In order words, you wanted the quantity:

$$\text{vol}\{(x_1, x_2) : \sum_{i=1}^{2} x_i^2 \leq 1\}$$

There is an almost universally known solution to this problem. This is just the area of the unit circle, which has been known for thousands of years as π. Even if I asked you to compute the volume of a 3-dimensional unit sphere, you would use the common formula $V = \frac{4}{3}\pi r^3$ to compute $\frac{4}{3}\pi$ (Figure 1).

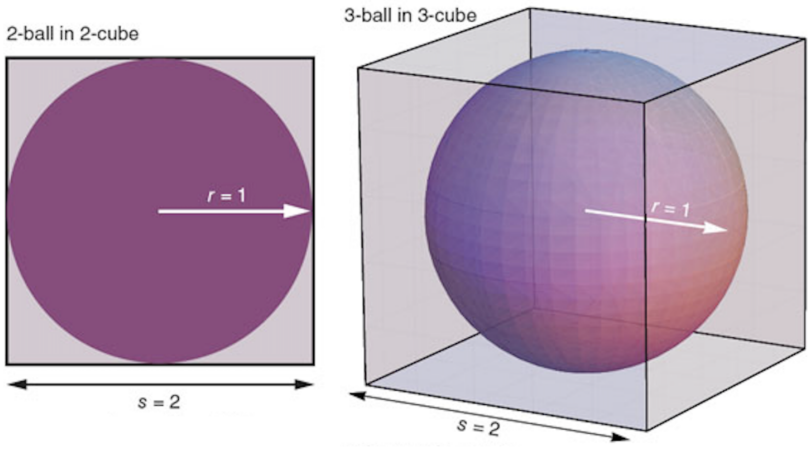


Figure 1: There exist closed form solutions for volume of 2, 3, and N dimensional unit balls

Consider increasing the dimension space. Instead, I ask you to compute the volume of the 4-D unit ball. I can't visualize that, and neither can you, but there exists a closed form solution using a function known as the Gamma Function (which will be relevant for verifying our method later). Using this, you could compute an exact solution to this task.

$$\Gamma(z) = \int_0^\infty t^{z-1} e^{-t}\, dt \qquad\qquad \mathcal{R}(z) > 0$$

$$V_n(R) = \frac{\pi^{n/2}}{\Gamma(\frac{n}{2}+1)}$$

Even if I ask you to compute 100 dimensions, you can use the gamma function and return an exact answer.

Now suppose that we want the following similar quantity:

$$\text{vol}\{(x_1, x_2, \ldots, x_{10}) : \sum_{i=1}^{10} x_i^4 \leq 1\}$$

Perhaps we don't want to deal with standard squared Euclidean space, or you're more constrained on the space you have to fill. Unfortunately, no clever gamma function exists for a similar computation, and nobody knows of a closed form solution.

But instead of an exact solution, what if I told you that the volume can be computed to within 0.2% of accuracy in seconds, just using standard software freely available to all computers, and that accuracy can be easily improved with faster hardware. For most practical, real-world applications, our approximation is more than sufficient. This is the motivation behind the Monte Carlo method.

2 Monte Carlo Integration

2.1 History

Stanislaw Ulam co-developed the general Monte-Carlo algorithm in the late 1940s, alongside Von Neumann and Nicholas Metropolis. The three were working in Los Alamos National Laboratory for the Manhattan Project during World War II, investigating the behavior of neutrons in the fissile material in atomic bombs. The problem at hand dealt with many known parameters in a complicated multi-dimensional relationship.

The scientists had all of the parameters known, but the equation was just too complicated to solve closed-form using standard methods at the time. After trying and repeatedly failing to calculate the result, Ulam thought of a new approach. Instead of looking for a closed-form solution, why not let the experiment play our many times, gathering data on-site. A fan of Solitaire, Ulam developed this method when thinking about the probabilities of complicated outcomes in a game. Rather than trying to predict the result using many different parameters from the game, why not just run the game many times and compute the likelihood that the specified outcome occurs? Given enough tests, he figured, this method would likely yield a nearly accurate result.

This method was named Monte Carlo after the famous gambling spot in Monaco, where Ulam's uncle would borrow money from the family and gamble it away. At the time information about the Manhattan Project was secret, so the three scientists published the first unclassified paper titled "The Monte Carlo Method" in 1949.

The development of this method coincided with the invention of modern electronic computers, which drastically sped up the computations of repetitive, numerical tasks. Soon after the initial publication, these methods were used in scientific fields all around Los Alamos, including in physical chemistry and the construction of the hydrogen bomb.

2.2 Overview

The central idea is as follows:

Suppose you were given a complicated domain $\Omega \subseteq \mathbb{R}^d$ and a complicated function $f : \Omega \to \mathbb{R}$, and you were asked to compute

$$\int_{\Omega} f(x)\ d\mathbf{x}$$

Given that the problem is deterministic, we know that there exists a solution to our function. But either f or Ω is too complicated, and we don't have a "nice", closed-form mathematical solution.

Monte Carlo aims to generate an approximation to this integral using repeated random sampling from the domain or a similar domain. As the sample size increases, the approximation converges to the real value of the function.

It is worth noting that other techniques for integral approximations exist. Such standard techniques include the Midpoint Rule, Trapezoidal Rule, and Simpson's Rule. Though it will not be proven formally here, the computational cost associated with these algorithms grows exponentially with the dimension space of the problem. Such an issue is referred to as the *curse of dimensionality* (Figure 2). As we will see shortly, Monte Carlo methods do not suffer from this problem by design of the random sampling of the algorithm. This advantage cannot be understated; in the real world a dimension can correspond to a degree of freedom, and so problems can consist of hundreds or thousands of dimensions.

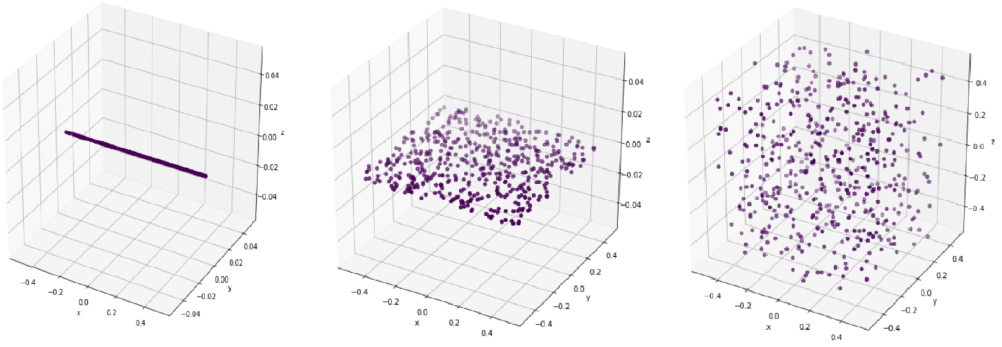


Figure 2: Curse of Dimensionality: Computational Cost grows exponentially with the dimension space

2.3 Applications

Our motivating example problem is still a relatively "nice", well-defined mathematical object.

Oftentimes in fields that deal with real-world phenomena (physics, chemistry, physics, stock markets, psychology) the function and domain are too complicated, and the integral simply cannot be solved in closed form. In practice this technique is the only known method that works for these computations.

2.4 Method

Monte-Carlo methods do not refer to one specific algorithm, but encompass the general idea of using random sampling to obtain a numerical approximation to a problem that is deterministic in nature. As such, there is not one standard algorithm, but here are two common methods for integral approximations (Figure 3). Let N denote the number of samples.

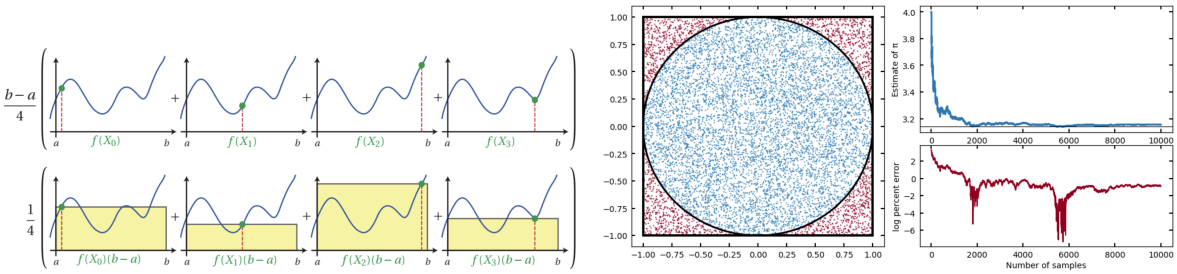
1. Crude Monte-Carlo Method
   (a) For N times: Randomly draw elements $x_i \in \Omega$ uniformly
   (b) Compute $\sum_{i=1}^{N} f(x_i)$, multiply by the domain ($b-a$ if 1-D integration on $x \in [a, b]$), and divide by the number of samples N
2. Geometric Esimation Method
   (a) Define a domain D of possible inputs with an easily computable volume such that the domain of interest is bounded
   (b) For N times: Randomly draw elements $x_i \in D$ uniformly
   (c) Compute the ratio of elements that were in Ω, and multiply that by the known volume V_D

3 Proof

Given $\Omega \subseteq \mathbb{R}^d$ and $f : \Omega \to \mathbb{R}$, we want to compute

$$\int_{\Omega} f \, d\mathbf{x}$$

Let us just consider $\Omega : [0, 1]^d$. The average value of $f : \Omega \to \mathbb{R}$ on Ω



(a) Crude Monte-Carlo (b) Geometric Estimation

Figure 3: 2 Types of Monte-Carlo Integration

$$\text{vol}(\Omega) = (1-0)^d = 1$$

$$f_{ave} = \frac{1}{|\Omega|}\int_{\Omega} f \; d\mathbf{x} = \frac{1}{(1-0)^d}\int_{[0,1]^d} f \; d\mathbf{x} = \int_{[0,1]^d} f \; d\mathbf{x}$$

We can approximate the average value of the function by summing up the values taken over N samples, then dividing by N:

$$\int_{[0,1]^d} f \; d\mathbf{x} \approx \frac{1}{n}\sum_{k=1}^{n} f(x_k)$$

$$d\mathbf{x} = dx_1 dx_2 \dots$$

Theorem 1. *Suppose $f : [0,1]^d \to \mathbb{R}$ is bounded. Then, if we define*

$$E(x_1, \dots, x_n) = \int_{[0,1]^d} f(x) \; d\boldsymbol{x} - \frac{1}{n}\sum_{k=1}^{n} f(x_k)$$

(i) Average value of $E = 0$

(ii) Average value of $|E|$ is

$$\frac{1}{\sqrt{n}}\left(\int_{[0,1]^d} (f - f_{ave}^2) \; dx\right)^{1/2}$$

Proof. (i) Average value of E = 0

Let $Q : [0,1] \times [0,1] \times \ldots \times [0,1] \to \mathbb{R}$ for each $x_i \in [0,1]$ in our domain. Thus, we get an N-dimensional domain:

Then,

$$Q : [0,1]^n \to \mathbb{R}$$

We define

$$Q(x_1, x_2, \ldots, x_n) = \frac{1}{n} \sum_{k=1}^{n} f(x_k)$$

to be the average value of f.

We will find the average value of Q

$$\begin{aligned}
Q_{ave} &= \frac{1}{\text{vol}[0,1]^n} \int_{[0,1]^n} Q(x_1, x_2, \ldots, x_n)\ d\mathbf{x} && \text{Def. of Average} \\
&= \int_{[0,1]^n} Q(x_1, x_2, \ldots, x_n)\ d\mathbf{x} && \text{vol}[0,1]^n = (1-0)^n = 1 \\
&= \int_{[0,1]^n} \frac{1}{n} \sum_{k=1}^{n} f(x_k) && \text{Def. of Q} \\
&= \frac{1}{n} \sum_{k=1}^{n} \int_{[0,1]^n} f(x_k) && \\
&= \frac{1}{n} \sum_{k=1}^{n} \int_0^1 \int_0^1 \ldots \int_0^1 f(x_k) dx_1 dx_2 \ldots dx_n && \text{Expanded} \\
&= \frac{1}{n} \sum_{k=1}^{n} \int_0^1 f(x_k) dx_k && \text{Fubini, } f(x_k) \text{ only depends on } x_k \\
&= \int_0^1 f(x)\ dx && \text{Sum up n times, divide by n}
\end{aligned}$$

Since x_k was arbitrary, we have,

$$Q_{ave} = \int_0^1 f(x)\ dx$$

so Q is an unbiased estimator of the integral

□

Proof. (ii) Average value of $|E|$ is

$$\frac{1}{\sqrt{n}}\left(\int_{[0,1]^d}(f-f_{ave}^2)\,dx\right)^{1/2}$$

Define

$$g(x)=f(x)-\int_{[0,1]^d}f(x)\,dx$$

so g is the deviation between f and f_{ave} at each point x

Now, we are imagining every sample is in d-dimensions, $[0,1]^d$ rather than $[0,1]$

Let's compute the average value of g^2

$$\left(\frac{1}{n}\sum_{k=1}^{n}g(x_k)^2\right)$$

We have

$$\frac{1}{(\text{vol}[0,1]^d)^n}\int_{[0,1]^{nd}}\left(\frac{1}{n}\sum_{k=1}^{n}g(x_k)^2\right)\,d\mathbf{x}$$

$$=\frac{1}{\text{vol}[0,1]^{nd}}\int_{[0,1]^{nd}}\left(\frac{1}{n}\sum_{k=1}^{n}g(x_k)^2\right)\,d\mathbf{x}$$

$$=\int_{[0,1]^{nd}}\left(\frac{1}{n}\sum_{k=1}^{n}g(x_k)^2\right)\,d\mathbf{x} \qquad \text{vol}[0,1]^n=1$$

$$=\int_{[0,1]^d}\int_{[0,1]^d}\cdots\int_{[0,1]^d}\left(\frac{1}{n}\sum_{k=1}^{n}g(x_k)^2\right)\,d\mathbf{x} \qquad \text{Expanded}$$

$$=\int_{[0,1]^d}\int_{[0,1]^d}\cdots\int_{[0,1]^d}\frac{1}{n^2}\sum_{k=1,l=1}^{n}g(x_k)g(x_l)\,d\mathbf{x}$$

$$=\frac{1}{n^2}\int_{[0,1]^d}\int_{[0,1]^d}\cdots\int_{[0,1]^d}\left(\sum_{k=1}^{n}g(x_k)^2+\sum_{k,l=1,k\neq l}^{n}g(x_k)g(x_l)\right)dx_1dx_2\ldots dx_n$$

Now, we can break this result into **(1)** and **(2)**

(1)

$$\frac{1}{n^2}\int_{[0,1]^d}\int_{[0,1]^d}\dots\int_{[0,1]^d}\sum_{k=1}^{n} g(x_k)^2 d\mathbf{x}$$

$$= \frac{1}{n^2}\sum_{k=1}^{n}\int_{[0,1]^d}\int_{[0,1]^d}\dots\int_{[0,1]^d} g(x_k)^2 dx_1 dx_2 \dots dx_n$$

$$= \frac{1}{n^2}\sum_{k=1}^{n}\int_{[0,1]^d} g(x_k)^2\ dx_k \qquad \text{Fubini, } f(x_k) \text{ only depends on } x_k$$

$$= \frac{1}{n}\int_{[0,1]^d} g(x_k)^2\ dx_k \qquad \text{Sum up n times, divide by } n^2$$

As $n \to \infty$, this gets closer to 0

(2)

$$\frac{1}{n^2}\int_{[0,1]^d}\int_{[0,1]^d}\dots\int_{[0,1]^d}\sum_{k,l=1,k\neq l}^{n} g(x_k)g(x_l)\ dx_1 dx_2 \dots dx_n$$

$$= \frac{1}{n^2}\sum_{k,l=1,k\neq l}^{n}\int_{[0,1]^d}\int_{[0,1]^d}\dots\int_{[0,1]^d} g(x_k)g(x_l)\ dx_1 dx_2 \dots dx_n$$

$$= \frac{1}{n^2}\sum_{k,l=1,k\neq l}^{n}\int_{[0,1]^d}\int_{[0,1]^d} g(x_k)g(x_l)\ dx_k dx_l \qquad \text{Only 2 integrals are relevant}$$

$$= \frac{1}{n^2}\sum_{k,l=1,k\neq l}^{n}\left(\int_{[0,1]^d} g(x_k)\ dx_k\right)\left(\int_{[0,1]^d} g(x_l)\ dx_l\right) \qquad \text{Split up the integral}$$

$$= 0$$

because when $k \neq l$, at least one of the integrals evaluate to 0.

Readers with a more proficient background in probability theory can see that we've shown that the covariance of any arbitrary pair of variables is 0.

Thus,

$$\left(\frac{1}{n}\sum_{k=1}^{n} g(x_k)^2\right) = \frac{1}{n^2}\int_{[0,1]^d}\int_{[0,1]^d}\dots\int_{[0,1]^d}\left(\sum_{k=1}^{n} g(x_k)^2 + \sum_{k,l=1,k\neq l}^{n} g(x_k)g(x_l)\right)dx_1dx_2\dots dx_n$$
$$= \frac{1}{n}\int_{[0,1]^d} g(x_k)^2\ dx_k$$

So,

$$E_{ave}^2 = \frac{1}{n}\int_{[0,1]^d}(f - f_{ave}^2)\ dx$$
$$|E|_{ave} = \frac{1}{\sqrt{n}}\left(\int_{[0,1]^d}(f - f_{ave}^2)\ dx\right)^{1/2}$$

□

4 Example

Here we will use the Monte Carlo integration technique, specifically the geometric estimation method, to estimate the volume of the 10 dimensional unit ball.

On each iteration, we choose 10 points in $[0, 1]$ uniformly at random, and combine them for our vector $\mathbf{x} \in \mathbb{R}^{10}$. If this vector is in the 10-D unit ball, ie. its points satisfy

$$\mathbf{x} = (x_1, x_2, \dots, x_{10}) : \sum_{i=1}^{10} \mathbf{x}_i^2 \leq 1$$

then we count it as in the shape. After repeating this process 1,000,000 times, we check the ratio of samples that are in the ball, and multiply by the volume of the 10D unit cube (which we know is 1) to obtain an estimate for the volume. Here is the code for the algorithm in Python:

```python
## Monte-Carlo Integration to Compute the Volume of the 10-D Unit Ball

import random as r

total_samples = 1000000 # One million total samples
expected_volumes = [0] * 10 # Empty array of 10 0's
num_of_repetitions = 10

# The box we're bounding ranges from -1 to 1 in all 10 dimensions
volume_of_box = (1 - -1)**10

for j in range(num_of_repetitions):

  num_in_shape = 0

  for i in range(total_samples):

    # Generates a random x_1, ..., x_10
    ten_dimensions = [0] * 10

    for k in range(10):
      x_k = (r.random() * 2) - 1
      ten_dimensions[k] = x_k

    # Checks whether the random numbers satisfy the property
    # of the set. If so, then adds the total count in num_of_shape

    if (sum_of_squares(ten_dimensions) <= 1):
      num_in_shape = num_in_shape + 1

  ratio_in_set = num_in_shape / total_samples
  expected_volume = ratio_in_set * volume_of_box

  expected_volumes[j] = expected_volume
```

We ran that 10 times, and got the following estimates for the volume:

$$\begin{aligned}\text{Expected Volumes} = [&2.62656, 2.601984, 2.57024, 2.509824, 2.573312,\\ &2.48832, 2.49856, 2.540544, 2.638848, 2.50386]\end{aligned}$$

We can compute the average of our predictions.

$$\begin{aligned}\bar{x} &= \frac{1}{N}\sum_{i=1}^{N} x_i \\ &= \frac{1}{10}\big(2.62656 + \ldots + 2.50386\big) \\ &= 2.55518\end{aligned}$$

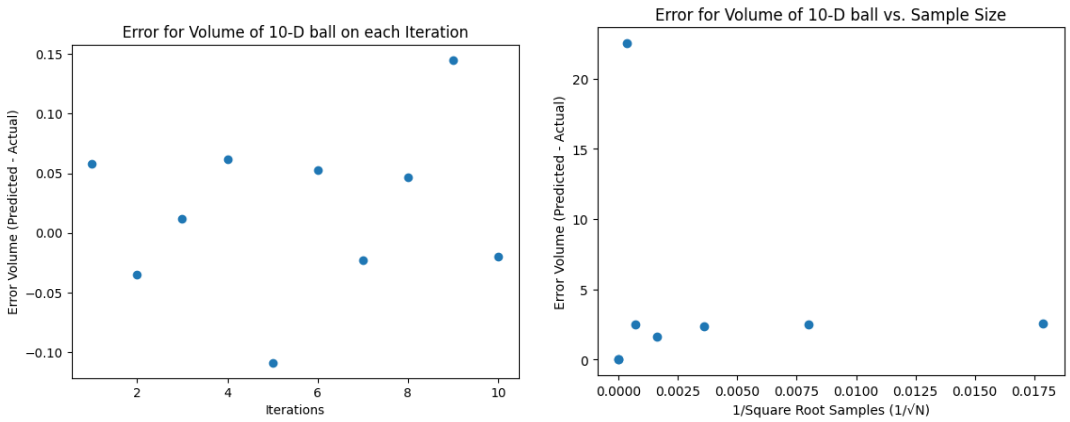
Using the Gamma Function, we know the exact solution is

$$V = \frac{\pi^5}{120} = 2.55016$$

Thus, we can compute the percent error of our measurement

$$\begin{aligned} \%\ Error &= \frac{Experimental - Theoretical}{Theoretical} * 100\% \\ &= \frac{2.55518 - 2.55016}{2.55016} * 100\% \\ &= 0.20342\% \end{aligned}$$

Thus, our estimate was 0.2% off. To increase accuracy, we could use a computer with higher performance, or wait for longer to sample more samples (as we've shown to eventually converge to the actual solution). We can also see visually the two statements we proved above (that the average $E = 0$, and that the average value of E^2 α $\frac{1}{\sqrt{N}}$ (Figure 4).



(a) Error appears to be centered on 0

(b) Apart from really small N, $|E|$ = const. * $\frac{1}{\sqrt{N}}$

Figure 4: Error for Monte-Carlo Integration Example

5 Markov-Chain Monte Carlo Methods

5.1 Motivation – Importance Sampling

For the rest of this paper, we will work with probability distributions. Suppose we want to calculate the expected value of an unknown probability distribu-

tion, which is closely related to computing integrals over the domain. To do so, we are going to need to generate tons of random samples.

The standard method to do this is called rejection sampling. Similar to the idea of geometric estimation, it involves bounding the distribution of interest with a proposed distribution of known size. We then take independent samples from the proposed distribution. If our sample is under the distribution of interest, we accept the sample in the distribution; if not, we reject (Figure 5).

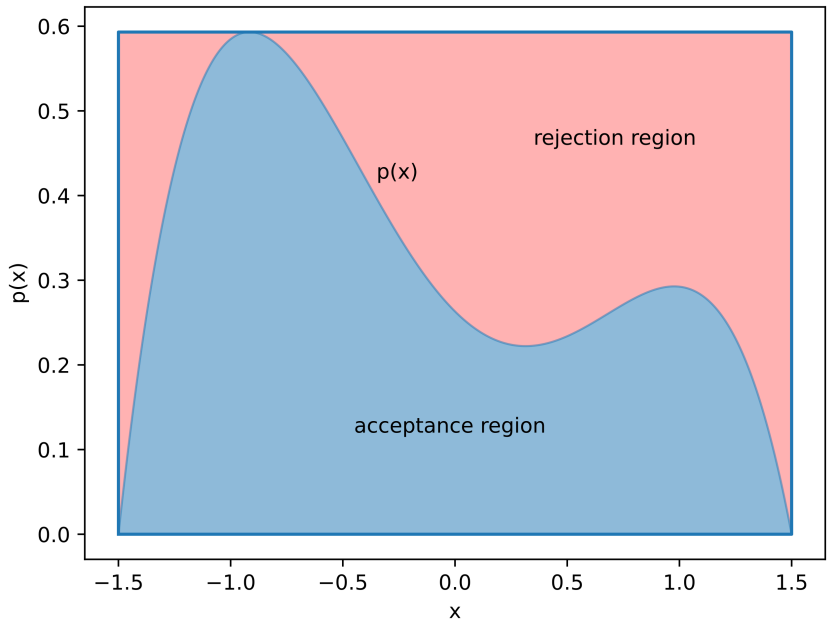


Figure 5: Rejection Sampling over Unknown Probability Distribution P(x)

Traditional Monte Carlo methods, like the example highlighted above, deal with uncorrelated samples. Each sample is independent and independently distributed, which has its benefits and drawbacks. It can be helpful that each sample is i.i.d. because samples taken without prior assumptions are crucial to remove bias from any statistical study. It can also potentially lead to a higher rate of convergence and degree of confidence, since we know that our method generalizes to a wide variety of inputs.

However, i.i.d. can also be problematic if the sample space has vast regions of 0 or near low probability. If such is the case, many of our generated random samples are wasted, because our sample did not land within the probability distribution we want to investigate.

Instead, we might want to learn to base our next sample based on the previous

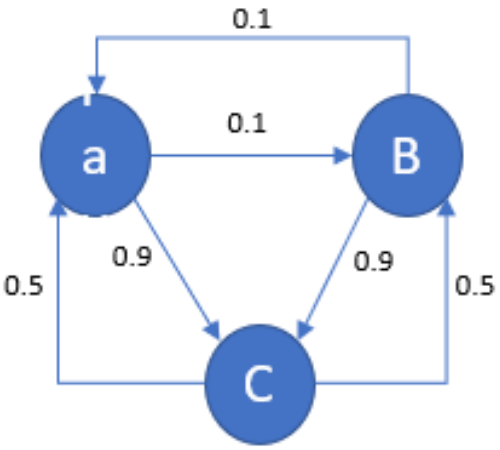
example, so that over time we get samples that are more likely to be within high density regions of the probability curve. That idea of basing our next sample on the previous (and only the prior step) ties directly into the idea of a Markov chain, which I'll briefly elaborate in the next subsection.

5.2 Markov Chains

Here we will briefly review Markov Chains, focusing on stationary distributions which are particularly relevant for the Metropolis-Hastings Algorithm.

Definition 1. *A discrete-time stochastic process (DTSP) is a sequence of random variables* $X^{(0)}, X^{(1)}, \ldots$ *where* $X^{(t)}$ *is the value at time* t

Definition 2. *A Markov Chain is special type of DTSP in which the probability of each event depends only on the state attained in the previous event*



$$P = \begin{bmatrix} 0 & 0.1 & 0.9 \\ 0.1 & 0.0 & 0.9 \\ 0.5 & 0.5 & 0.0 \end{bmatrix}$$

Figure 6: Example Markov Chain

We can define a row vector to represent the probability that we are in each state in the chain at time t.

$$q^{(t)} = [q_a^{(t)}, q_b^{(t)}, q_c^{(t)}]$$

Then, from above, we can define a "transition probability matrix" P to store the probabilities of transitioning from one state to another, given the values in the previous state

Thus,

$$
\begin{aligned}
q^{(1)} &= q^{(0)}P \\
q^{(2)} &= q^{(1)}P = (q^{(0)}P)P = q^{(0)}P^2 \\
&\dots
\end{aligned}
$$

And we can derive by induction that

$$q^{(t)} = q^{(0)}P^t \qquad \text{for all } \ t \geq 0$$

Next, we introduce the idea of a stationary distribution, which is essential for understanding the mechanisms of the Metropolis-Hastings algorithm.

Since a Markov Chain is deterministic, and only depends on the immediate prior value, not that if $q^{(t+1)} = q^{(t)}$, the Markov chain will never change again. We call that a stationary distribution.

Definition 3. *A stationary distribution π is a row vector, whose entries are non-negative and sum to 1, is unchanged by the operation of transition matrix P on it. Formally,*

$$\pi P = \pi \qquad \sum_{i=1}^{n} \pi_i = 1$$

There is a theorem that states that in the long run, any irreducible, aperiodic Markov Chain converges to its unique stationary distribution. The proof of this statement is beyond the scope of this paper.

The last concept that will serve useful for the mechanisms of the Metropolis-Hastings Algorithm is the detailed balance condition.

Definition 4. *Let $X^{(0)}, X^{(1)}, \dots$ be a Markov Chain with stationary distribution π. The chain satisfies detailed balance if $\forall i, j$*

$$\pi_i p_{i,j} = \pi_j p_{j,i}$$

Theorem 2. *Let X_n be a Markov Chain, and let P be the transition probability matrix. Suppose there exists a distribution π that satisfies the detailed balance condition. Then π is a stationary distribution of the chain.*

Proof. Given that π satisfies detailed balance, we can write

$$\sum_i \pi_i p_{i,j} = \sum_i \pi_j p_{j,i} = \pi_j \sum_i p_{i,j} = \pi_j$$

Thus, $\pi = \pi P$, and so π is a stationary distribution. □

6 Metropolis-Hastings Algorithm

6.1 History

After the popularization of the general Monte-Carlo algorithm, and fueled by the power of electronic computing, many scientists were using random sampling to estimate unknown quantities.

Part of the process of Monte-Carlo methods involves generating random samples from a domain of interest. Consider the challenge of sampling from an unknown probability distribution. The traditional methods at the time used rejection sampling, which could potentially take many samples before finding one of interest. To get more samples quicker required some form of importance sampling.

After World War II, Nicholas Metropolis was back to Los Alamos. In 1953, his research group, composed of Marshall and Arianna Rosenbluth, Edward Teller, and August Teller, published a paper titled *Equation of State Calculations by Fast Computing Machines* that provided one such algorithm to sample from probability distributions using importance sampling. Known today as the Metropolis-Hastings algorithm, it provided a general method to simulate observations from virtually any distribution.

The Metropolis Algorithm uses a symmetrical distribution to generate new samples from previous ones. Wilfred Keith Hastings generalized this algorithm with the possibility of non-symmetric jumping distributions in his 1970 paper titled *Monte Carlo Sampling Methods Using Markov Chains and Their*

Applications. His contribution provides a the modern method to sample from a high-dimensional probability distribution.

Though credited with the algorithm, there is controversy about how much Metropolis himself actually contributed to the algorithm. Metropolis was the one who co-developed the Monte-Carlo method alongside Stanislaw Ulam and Von Neumann. However until 2003, little was know about the algorithm's actual development. Marshall Rosenbluth denies that Metropolis was the founder of the algorithm. Instead, he credits Edward Teller for proposing the problem, himself with solving it, and Arianna with programming the computer, with Metropolis only necessary for testing the method. This contrasts with Metropolis' story and Edward's memoir, where he writes that the five of them worked together for days and nights crafting the algorithm.

6.2 Overview

Below I will outline the general overview of the problem, followed by the derivation for how to choose the next sample from the previous following the detailed balance condition. We won't provide a formal, rigorous proof why this converges, but I'll conclude with some intuition behind the design of the algorithm and why it works.

Let $P(x)$ be the probability distribution that we want to draw samples from. Let $f(x)$ be a known function proportional to the density P (with unknown normalization constant).

Let $g(x \mid y)$ be a given probability density function centered at y, often called the proposal density or jumping distribution. A frequent choice of $g(x \mid y)$ is the Gaussian distribution centered at y with chosen sigma. Note: g does not necessarily have to be symmetric (it was Hastings who formed this generalization).

The Metropolis-Hastings algorithm works as follows:

1. Initialization
   (a) Choose an initial arbitrary sample x_0 and set $t = 0$
2. Let x_t be the given sample at time t. For each iteration:
   (a) Choose a candidate sample x' by sampling randomly on $g(x' \mid x_t)$
   (b) Calculate the acceptance probability:

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

(c) Let $u \in [0, 1]$ be a uniform random number

(d) If

$$\begin{cases} u \leq A & \text{Accept; } x_{t+1} = x' \\ u > A & \text{Reject; } x_{t+1} = x_t \end{cases}$$

(e) Increment time step, $t = t + 1$

As we continue to increment time, the Metropolis-Hastings algorithm generates a Markov Chain of possible samples from $P(x)$. As we take more samples, the distribution more closely approximates the desired distribution. From the convergence of importance sampling, this Markov Chain has the desired distribution at equilibrium for some timestep t. When a Markov Chain converges to a stationary distribution, it will revisit the states in accordance with their probability. Therefore, we can continue to increment the process and choose representative samples for times $t^* \geq t$.

Where does A come from?

If we can show that the probability of going to any state x_t to x_{t+1} satisfies the detailed balance condition, then we have already shown that $P(x)$ is a stationary distribution, and so the Markov Chain will be sampling from $P(x)$.

We will show how our choice of A satisfies the detailed balance condition.

$$P(x' \mid x)P(x) = P(x \mid x')P(x') \qquad \text{Detailed Balance}$$

$$\frac{P(x' \mid x)}{P(x \mid x')} = \frac{P(x')}{P(x)} \tag{1}$$

The probability of transitioning to the next state consists of two steps: proposal + acceptance.

Let $g(x' \mid x)$ be the proposal distribution (the conditional probability of proposing a state x' given x).

Let $A(x', x)$ be the acceptance distribution (the probability to accept state x').

Then we can write,

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

and by equation (1),

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

Note that the acceptance ratio ($\alpha = \frac{A(x',x)}{A(x,x')}$) indicates how probable the new proposed sample is with respect to the current sample, according to the distribution with density $P(x)$. If the proposed sample is more probable than the existing point (i.e. it lies on a point of higher density in the distribution $P(x)$), we'd want to always accept that value. In that case, $\alpha > 1 \geq u$.

$$\text{If } \alpha = \frac{A(x', x)}{A(x, x')} = \frac{P(x')}{P(x)} \times \frac{g(x \mid x')}{g(x' \mid x)} > 1, \text{ accept}$$

We don't want to discount less probable points, but just make the probability of acceptance proportional to the relative densities of the region. Thus, if $\alpha < 1$, we should accept with probability α, and reject with probability $1 - \alpha$. By doing so, we tend to stay in (and return a larger number of samples from) high-density regions of $P(x)$, while only occasionally sampling from low-density regions.

$$\text{If } \alpha = \frac{A(x', x)}{A(x, x')} = \frac{P(x')}{P(x)} \times \frac{g(x \mid x')}{g(x' \mid x)} < 1, \text{ accept w/ prob. } \alpha \text{ and reject w/ prob. } 1 - \alpha$$

These two conditions can be summarized in the following statement:

$$\text{Probability of Acceptance} = \min\left(1, \frac{P(x')}{P(x_t)} \frac{g(x_t \mid x')}{g(x' \mid x_t)}\right)$$

6.3 Analysis

Compared with standard acceptance/rejection sampling, the Metropolis-Hastings algorithm offers significant advantages. Correlated sampling serves as the crux of these improvements. In standard rejection sampling, since samples are i.i.d. we can potentially waste many samples that are taken from regions or small or negligible density. Using correlated samples allows for the algorithm to learn from its previous successes and failures, resulting in samples that converge much quicker to samples from the unknown probability distribution. Then, though this algorithm can still potentially suffer from the *curse of dimensionality* given the exponentially shrinking likelihood of choosing proper samples,

learning from previous attempts can potentially dampen its effects over the long term.

However, there are a few drawbacks to this algorithm that are worth noting. One is that given that the samples are uncorrelated, even though in the long term the samples do reflect $P(x)$, a subset of nearby samples are often much too intertwined to be thought of as individual samples, and can potentially lead to larger errors when viewed on a small scale. Another problem is that though the Markov chain eventually converges to the desired distribution over the long run, initial samples may follow a very different distribution (especially if the starting point is a region of low density). It is therefore crucial to select a strong starting candidate x_0, as well as the jumping distribution g. Commonly, simulations will run this sample method many many times, and throw away the first samples as the chain does not yet reflect the distribution of interest (called a burn-in-period).

7 Metropolis-Hastings Example

Here we will use the Metropolis-Hastings Example to compute the expected value of an unknown distribution, given the known distribution:

$$f(x) = e^{-(x+1)^2(x-1)^2}$$

In this analysis, we will compare the M-H algorithm with the standard rejection sampling method, to understand how different algorithms yield different accuracies, convergence rates, and correlations.

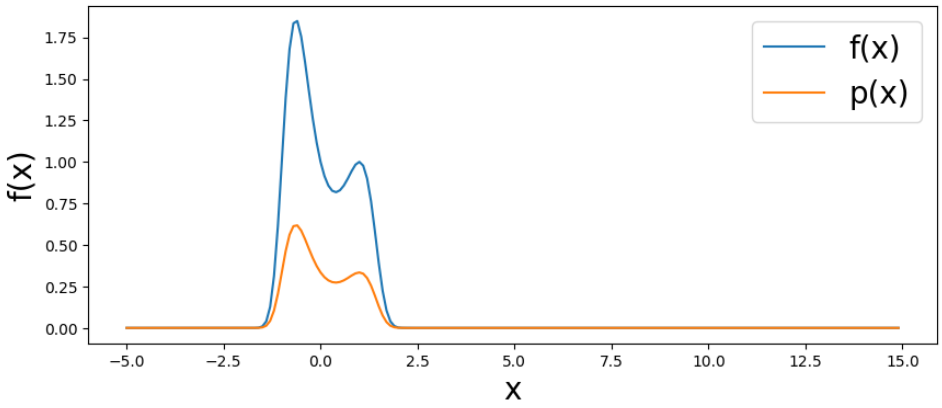


Figure 7: Unknown Distribution P(x) vs. Known Distribution f(x)

First, we graph our known distribution $f(x)$ on top of our probability density function of interest, $P(x)$ (Figure 7). Using Wolfram Alpha for verification purposes, we have that

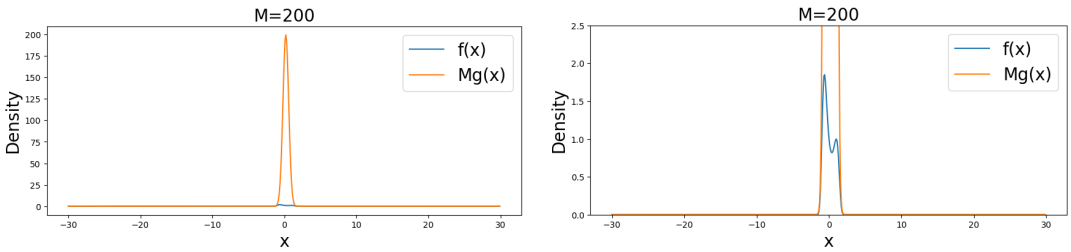
$$\int_{-\infty}^{\infty} f(x)\ dx \approx 2.9936$$

and hence that becomes our normalization factor (obtaining the normalization factor is just for analysis purposes, our simulation will not know that).

Using standard rejection sampling, we find that using a Gaussian distribution

$$g(x) \sim \mathcal{N}(\mu = 0.2, \sigma = 0.4)$$

scaled up by a factor of 200 covers our distribution of interest well-enough to draw samples from (Figure 8).

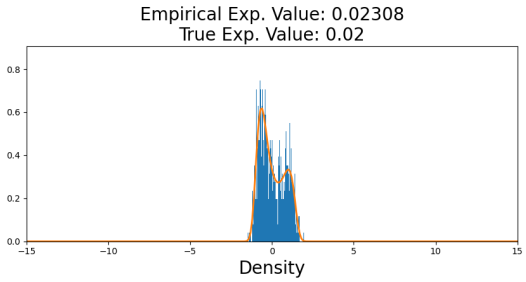


(a) (b)

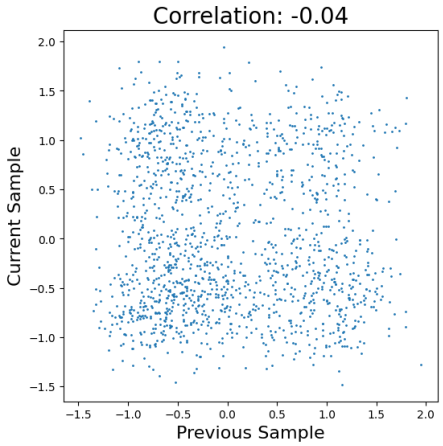
Figure 8: Scaled up Outer Distribution to Sample From

After taking 100,000 uniform samples from 200g(x), we find that only 1545 samples actually fell within P(x) (efficiency = 0.015). Our computed mean using those samples is 0.01837, compared to the true distribution mean of 0.02151. Our correlation of data points is 0.04, near 0 as our samples are i.i.d.

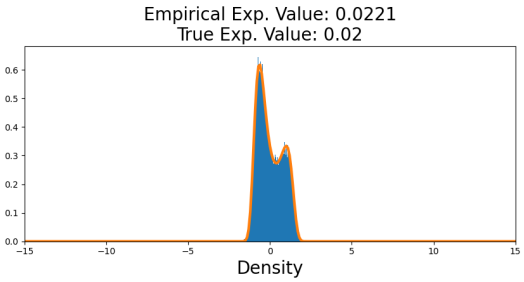
We then run the same simulation using the Metropolis-Hastings algorithm, with $g(x \mid y) \sim \mathcal{N}(y, 0.4)$. Throwing away the first 1000 of the 10000 samples yields an efficiency of 0.99. Our predicted expectation is 0.02841, much closer to the true value. We see that the distribution appears much less jagged than for rejection sampling. Our correlation of data points is 0.91, near 1 as each sample relies on the previous.



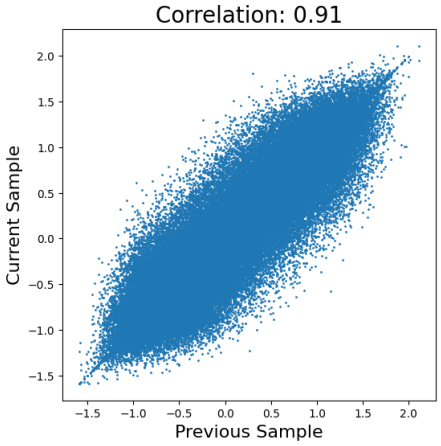
(a) Estimation w/ Rejection Sampling



(b) Correlation w/ Rejection Sampling



(c) Estimation w/ M-H Algorithm



(d) Correlation w/ M-H Algorithm

Figure 9: Rejection Sampling vs. Metropolis-Hastings Algorithm

Though the algorithm is still far from perfect, it provides evidence for a faster rate of convergence to the true mean value than conventional methods of sampling. The results of this algorithm can be transformative to fields where we rely on faster, more accurate numerical approximations to probability distributions.

8 References

Images:

1. https://www.americanscientist.org/article/an-adventure-in-the-nth-dimension
2. https://www.linkedin.com/pulse/curse-dimensionality-divya-gera/
3. https://www.scratchapixel.com/lessons/mathematics-physics-for-computer-graphics/monte-carlo-methods-in-practice/monte-carlo-integration.html
4. https://prappleizer.github.io/Tutorials/MetropolisHastings/MetropolisHastings_Tutorial.html
5. https://commons.wikimedia.org/wiki/File:Rejection_sampling_of_a_bounded_distribution_with_finite_support.svg
6. https://towardsdatascience.com/markov-chain-analysis-and-simulation-using-python-4507cee0b06e
7. https://commons.wikimedia.org/wiki/File:Metropolis_hastings_algorithm.png

Sources:

1. https://en.wikipedia.org/wiki/Monte_Carlo_method
2. https://en.wikipedia.org/wiki/Monte_Carlo_integration
3. https://towardsdatascience.com/the-basics-of-monte-carlo-integration-5fe16b40482d
4. https://prappleizer.github.io/Tutorials/MetropolisHastings/MetropolisHastings_Tutorial.html
5. https://victortuekam.com/posts/9
6. https://www.algorithm-archive.org/contents/metropolis/metropolis.html
7. https://www.youtube.com/@ritvikmath
8. https://courses.cs.washington.edu/courses/cse312/22au/
9. https://www.jstor.org/stable/pdf/30037292.pdf
10. https://thedecisionlab.com/reference-guide/statistics/monte-carlo-simulation