Bayesian inference and Monte Carlo methods

Lecture 3: Monte Carlo Methods

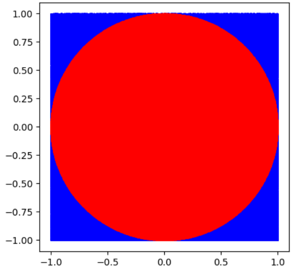
Susana J. Landau

25 de agosto de 2025

Introduction

- Monte Carlo methods provide approximate solutions to a wide variety of mathematical problems by performing experiments with samples of pseudo-random numbers on a computer. This method is applicable to any type of problem, whether stochastic or deterministic.
- The term ”Monte Carlorefers to the Monte Carlo casino, one of the capitals of games of chance, and was chosen for these techniques due to their inherent randomness.
- The method was developed by Stanislav Ulam and John Von Neumann.
- Unlike numerical methods that rely on evaluations at N points in an M-dimensional space to produce an approximate solution, the Monte Carlo method has an absolute estimation error that decreases as $\frac{1}{\sqrt{N}}$, according to the central limit theorem.

Calculation of π



- If we randomly throw needles onto the square, we can see that:

$$\frac{\text{number of needles landing inside the circle}}{\text{total number of needles}} = \frac{\text{Area of the Circle}}{\text{Area of the Square}} = \frac{\pi}{4}$$

- By performing N trials of this method, we obtain a sample of possible values for π.

Calculation of π

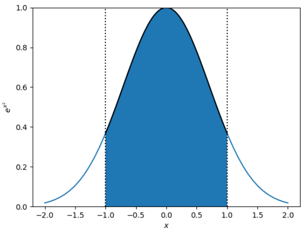
- The calculation of π is performed using a computational experiment with two possible outcomes. Therefore, we can estimate the error of this calculation as the standard deviation of the binomial distribution. Defining the number of hits as $\epsilon = \frac{N}{N_0}$, where N is the number of needles inside the circle and N_0 is the total number of needles:

$$\begin{aligned} \delta N &= \sigma = \sqrt{N_0\epsilon(1-\epsilon)} \\ \frac{\delta N}{N} &= \frac{\sqrt{N_0 N/N_0(1-\epsilon)}}{N} = \sqrt{\frac{1-\epsilon}{N}} \end{aligned}$$

- The more similar the area of the circle is to the area of the figure where the needles are thrown, the larger the number of hits N will be, and therefore the more precise the estimation of π.

Calculation of Integrals

- By applying a method similar to the one used for calculating π, we can compute integrals using Monte Carlo methods.



The model for this calculation is based on:

$$\int_a^b f(x)dx = \frac{(b-a)NF_{\max}}{N_0}$$

where
N is the number of points below the curve, $F_{\text{máx}}$ is the maximum value of the function to be integrated over the interval, and N_0 is the total number of points.

Calculation of Integrals with Markov Chains

- Instead of generating N random points in a square, we generate a Markov chain, where each point in the chain depends only on the previous step.
- The points are generated as follows:

$$\begin{aligned} x_{i+1} &= x_i + \Delta x \\ y_{i+1} &= y_i + \Delta y \end{aligned}$$

  where Δx and Δy are randomly generated numbers.
- At each step, we must check:

$$\begin{aligned} x_i &\in [a, b] \\ y_i &\in [0, F_{\text{máx}}] \end{aligned}$$

- If the previous condition is not satisfied, then $x_{i+1} = x_i$.

Calculation of Statistical Indicators with the Naive Monte Carlo Method

- Suppose we want to calculate the expected value of a function $f(x)$, where x is a random variable with probability $p(x)$:

$$E_p[x] = \int p(x) f(x) dx$$

Calculation of Statistical Indicators with the Naive Monte Carlo Method

- Suppose we want to calculate the expected value of a function $f(x)$, where x is a random variable with probability $p(x)$:

$$E_p[x] = \int p(x) f(x) dx$$

- We can see that:

$$E_p[x] = \int p(x) f(x) dx \sim \frac{1}{N} \sum_{i=1}^{N} f(x_i)$$

where N values x_i are chosen by sampling from the distribution $p(x)$.

Monte Carlo Method with Importance Sampling

- Suppose now that we want to calculate $\int p(x)f(x)dx$, but it is not easy to sample from or evaluate $p(x)$. However, we know another distribution $q(x)$ from which it is easy to sample and evaluate, and $q(x)$ is large where $p(x)f(x)$ is large. Then,

$$\int p(x)f(x)dx = \int q(x)\frac{p(x)}{q(x)}f(x)dx \sim \frac{1}{N}\sum_{i=1}^{N}\frac{p(x_i)}{q(x_i)}f(x_i)$$

  where N values x_i are chosen by sampling from the distribution $q(x)$.

Example

- We want to obtain the expected value of the function $f(x) = e^{-x+\cos(x)}$ over the interval $(0, \infty)$, where x is a random variable with probability e^{-x}.

$$\int_0^\infty e^{-x+\cos(x)} e^{-x} dx = \int_0^\infty e^{-2x+\cos(x)} dx \sim \sum_{i=1}^{N} e^{-x_i+\cos(x_i)}$$

where the sample x_i is obtained by sampling from the distribution e^{-x}.

Example

- We want to obtain the expected value of the function $f(x) = e^{-x+\cos(x)}$ over the interval $(0, \infty)$, where x is a random variable with probability e^{-x}.

$$\int_0^\infty e^{-x+\cos(x)} e^{-x} dx = \int_0^\infty e^{-2x+\cos(x)} dx \sim \sum_{i=1}^{N} e^{-x_i+\cos(x_i)}$$

  where the sample x_i is obtained by sampling from the distribution e^{-x}.

- With importance sampling, the calculation is performed as follows:

$$\int_0^\infty \frac{e^{-2x+\cos(x)}}{g(x)} g(x) dx \sim \sum_{i=1}^{N} \frac{e^{-2x_i+\cos(x_i)}}{g(x_i)}$$

  where the sample x_i is obtained by sampling from the distribution $g(x)$.

Example

- We want to obtain the expected value of the function $f(x) = e^{-x+\cos(x)}$ over the interval $(0, \infty)$, where x is a random variable with probability e^{-x}.

$$\int_0^\infty e^{-x+\cos(x)} e^{-x} dx = \int_0^\infty e^{-2x+\cos(x)} dx \sim \sum_{i=1}^{N} e^{-x_i+\cos(x_i)}$$

  where the sample x_i is obtained by sampling from the distribution e^{-x}.

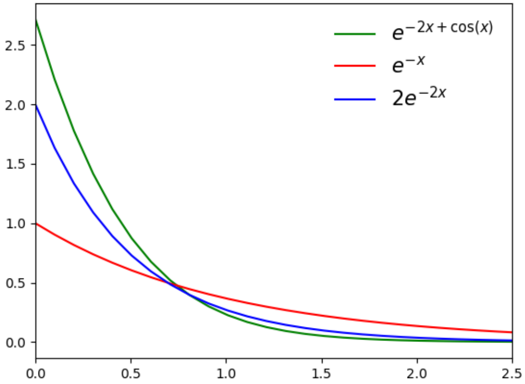
- With importance sampling, the calculation is performed as follows:

$$\int_0^\infty \frac{e^{-2x+\cos(x)}}{g(x)} g(x) dx \sim \sum_{i=1}^{N} \frac{e^{-2x_i+\cos(x_i)}}{g(x_i)}$$

  where the sample x_i is obtained by sampling from the distribution $g(x)$.

- What would be a good choice for $g(x)$?

Monte Carlo Method with Importance Sampling



Sampling from a Distribution

- The numpy library allows sampling from several predefined functions. For example:
- **numpy.random.uniform(a, b, N)**
  returns N random values from the uniform distribution $(p(x) = \frac{1}{b-a})$ in the interval $[a, b)$.
- **numpy.random.normal(mu, sigma, N)**
  returns N random values from the normal distribution with mean μ and standard deviation σ.
- **numpy.random.exponential(b, N)**
  returns N random values from the distribution $\frac{1}{b}e^{-\frac{x}{b}}$.

Sampling from a Distribution

- The numpy library allows sampling from several predefined functions. For example:
- **numpy.random.uniform(a, b, N)**
  returns N random values from the uniform distribution $(p(x) = \frac{1}{b-a})$ in the interval $[a, b)$.
- **numpy.random.normal(mu, sigma, N)**
  returns N random values from the normal distribution with mean μ and standard deviation σ.
- **numpy.random.exponential(b, N)**
  returns N random values from the distribution $\frac{1}{b}e^{-\frac{x}{b}}$.
- To sample from other distributions, an operation on the uniform distribution is performed and must be calculated case by case.

Generating Distributions Using Variable Transformation

- The problem consists of obtaining samples from the distribution $f(x)$ using samples from the uniform distribution $u(x)$.

$$\int_{-\infty}^{x} f(x')dx' = \int_{-\infty}^{r} u(r')dr' = r(x)$$

Generating Distributions Using Variable Transformation

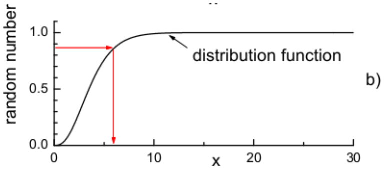
- The problem consists of obtaining samples from the distribution $f(x)$ using samples from the uniform distribution $u(x)$.

$$\int_{-\infty}^{x} f(x')dx' = \int_{-\infty}^{r} u(r')dr' = r(x)$$

- Recall that the cumulative distribution function (CDF) is $F(x) = \int_{-\infty}^{x} f(x')dx'$. Therefore:

$$\begin{aligned} F(x) &= U(r) = r \\ x(r) &= F^{-1}(r) \end{aligned}$$

- If r is a sample from a uniform distribution, it can be thought of as the result of applying F to a distribution x. You can find x by applying F^{-1} to r.



Generating Distributions Using Variable Transformation

- The method is as follows:
  - Generate a sample r from the uniform distribution.
  - Calculate $x = F^{-1}(r)$.
- The method can be applied to any distribution, but it is particularly useful for distributions that can be expressed in terms of their CDF.

Generating Distributions Using Variable Transformation

- $f(x) = 2x$ for $0 \leq x < 1$
  $x(r) = ?$

Generating Distributions Using Variable Transformation

- $f(x) = 2x$ for $0 \leq x < 1$
  $x(r) =?$
  $x(r) = \sqrt{r}$

Generating Distributions Using Variable Transformation

- $f(x) = 2x$ for $0 \leq x < 1$
  $x(r) = ?$
  $x(r) = \sqrt{r}$
- $f(x) = \gamma e^{-\gamma x}$
  $x(r) = ?$

Generating Distributions Using Variable Transformation

- $f(x) = 2x$ for $0 \leq x < 1$
  $x(r) = ?$
  $x(r) = \sqrt{r}$
- $f(x) = \gamma e^{-\gamma x}$
  $x(r) = ?$
  $x(r) = -\frac{1}{\gamma} \ln (1 - r)$

Generation of a Normal Distribution in the Variables x and y

- Consider the expression for a two-dimensional normal distribution:

$$f(x,y) = \frac{1}{2\pi} e^{-(x^2+y^2)/2}$$

Generation of a Normal Distribution in the Variables x and y

- Consider the expression for a two-dimensional normal distribution:

$$f(x,y) = \frac{1}{2\pi} e^{-(x^2+y^2)/2}$$

- A change of variables to polar coordinates can be performed:
  $x = r\cos\theta \qquad y = r\sin\theta \qquad \frac{\partial(x,y)}{\partial(r,\theta)} = r$
- This yields $g(r,\phi) = \frac{1}{2\pi} r e^{-r^2/2}$ and the marginal distributions:

$$\begin{aligned} g_r &= \int_0^{2\pi} g(r,\phi) d\phi = \int_0^{2\pi} \frac{1}{2\pi} r e^{-r^2/2} d\phi = r e^{-r^2/2} \\ g_\phi &= \int_0^{2\pi} g(r,\phi) dr = \int_0^{2\pi} \frac{1}{2\pi} r e^{-r^2/2} dr = \frac{1}{2\pi} \end{aligned}$$

Generation of a Normal Distribution in the Variables x and y

- Now we repeat the procedure used for the one-dimensional case for each of the marginal distributions:

$$\begin{aligned} \int_0^{\rho} \rho' e^{-\rho'/2} d\rho' = \int_{-\infty}^{r_1} u(r)dr &= r_1 \\ -e^{-\rho'/2}|_0^{\rho} &= r_1 \\ 1 - e^{-\rho/2} &= r_1 \\ \rho = \sqrt{-2\ln(1-r_1)} & \end{aligned}$$

and for the variable ϕ:

$$\begin{aligned} \int_0^{\phi} \frac{1}{2\pi} &= r_2 \\ \phi = 2\pi r_2 & \end{aligned}$$

Generation of a Normal Distribution in the Variables x and y

- Now we repeat the procedure used for the one-dimensional case for each of the marginal distributions:

$$\begin{aligned} \int_0^{\rho} \rho' e^{-\rho'/2} d\rho' = \int_{-\infty}^{r_1} u(r)dr &= r_1 \\ -e^{-\rho'/2}|_0^{\rho} &= r_1 \\ 1 - e^{-\rho/2} &= r_1 \\ \rho = \sqrt{-2\ln\left(1 - r_1\right)} & \end{aligned}$$

and for the variable ϕ:

$$\begin{aligned} \int_0^{\phi} \frac{1}{2\pi} &= r_2 \\ \phi = 2\pi r_2 & \end{aligned}$$

- In this way, we obtain:

$$\begin{aligned} x &= \sqrt{-2\ln\left(1 - r_1\right)}\cos\left(2\pi r_2\right) \\ y &= \sqrt{-2\ln\left(1 - r_1\right)}\sin\left(2\pi r_2\right) \end{aligned}$$

Method to Generate a Distribution $f(x)$ from a Uniform Distribution

- Choose $f(x)$ and the interval $[a, b]$.

Method to Generate a Distribution $f(x)$ from a Uniform Distribution

- Choose $f(x)$ and the interval $[a, b]$.
- Convert $f(x)$ into a probability distribution by calculating:

$$\int_a^b f(x)dx = F(b) - F(a) = c$$

Thus, the probability distribution is $\hat{f}(x) = \frac{f(x)}{c}$.

Method to Generate a Distribution $f(x)$ from a Uniform Distribution

- Choose $f(x)$ and the interval $[a, b]$.
- Convert $f(x)$ into a probability distribution by calculating:

$$\int_a^b f(x)dx = F(b) - F(a) = c$$

  Thus, the probability distribution is $\hat{f}(x) = \frac{f(x)}{c}$.
- Calculate x using the following expression:

$$\begin{aligned} \frac{F(x) - F(a)}{c} &= r \\ x = F^{-1}(F(a) + rc) \end{aligned}$$

Markov Chain Monte Carlo

- Given a joint probability distribution $p(\theta|D, I)$, where θ is a parameter, D is the data, and I is the prior information, to know the mean value of this distribution or its confidence intervals, we need to calculate integrals. For example, to calculate the mean value of a parameter θ:

$$\langle\theta\rangle = \int \theta\, p(\theta|D, I) d\theta = \int g(\theta) d\theta$$

  In many cases these integrals cannot be calculated analytically, and the numerical methods are time consuming.

Markov Chain Monte Carlo

- Given a joint probability distribution $p(\theta|D, I)$, where θ is a parameter, D is the data, and I is the prior information, to know the mean value of this distribution or its confidence intervals, we need to calculate integrals. For example, to calculate the mean value of a parameter θ:

$$\langle\theta\rangle = \int \theta\, p(\theta|D, I) d\theta = \int g(\theta) d\theta$$

In many cases these integrals cannot be calculated analytically, and the numerical methods are time consuming.

Markov Chain Monte Carlo (MCMC) methods are used to sample from the posterior distribution $p(\theta|D, I)$, allowing us to estimate the mean value and confidence intervals of the parameters.

Markov Chain Monte Carlo

- In straight Monte Carlo integration, the procedure is to pick n points, uniformly randomly distributed in a multi-dimensional volume (V) of our parameter space θ-. The volume must be large enough to contain all regions where $g(\theta)$ contributes significantly to the integral. Then the basic theorem of Monte Carlo integration estimates the integral of $g(\theta)$ over the volume V by
$$\int g(\theta)d\theta \approx \frac{V}{n}\sum_{i=1}^{n} g(\theta_i)$$
- One of the problems with straight Monte Carlo integration is that too much time is wasted sampling regions where $p(\theta|D,I)$ is very small.
- In general, drawing samples independently from $p(\theta|D,I)$ is not currently computationally feasible for problems where there are large numbers of parameters.
- However, the samples need not necessarily be independent. They can be generated by any process that generates samples from the target distribution, $p(\theta|D,I)$, in the correct proportions.

Markov Chain Monte Carlo

- All MCMC algorithms generate the desired samples by constructing a kind of random walk in the model parameter space such that the probability for being in a region of this space is proportional to the posterior density $p(\theta|D;I)$ for that region.
- The random walk is accomplished using a Markov chain, whereby the new sample, θ_{t+1}, depends on the previous sample θ_t according to an entity called the transition probability or transition kernel, $p(\theta_{t+1}|\theta_t)$. The transition kernel is assumed to be time independent.
- The remarkable property of $p(\theta_{t+1}|\theta_t)$ is that after an initial burn-in period (which is discarded) it generates samples of θ with a probability density equal to the desired posterior $p(\theta|D;I)$.

How does MCMC work?

- The MCMC process begins with an initial guess for the parameter θ_0.
- At each step t, a new candidate sample θ_{t+1} is selected from a proposal distribution, $q(\theta_{t+1}|\theta_t)$. (can be an arbitrary distribution, but usually a Gaussian with mean θ_t is used).
- The proposed sample is accepted or rejected based on a criterion (usually Metropolis-Hastings but there are others), which ensures that the samples are drawn from the correct posterior distribution $p(\theta|D;I)$.
- If the proposed sample is accepted, it becomes the new current sample; otherwise, the current sample is retained.
- This process is repeated for a large number of iterations, allowing the Markov chain to explore the parameter space and converge to the target distribution $p(\theta|D;I)$.

The Metropolis-Hastings algorithm

- Start with an initial guess θ_0.
- For each iteration t:
  1. Propose a new sample α_{t+1} from the proposal distribution $q(\alpha_{t+1}|\theta_t)$.

The Metropolis-Hastings algorithm

- Start with an initial guess θ_0.
- For each iteration t:
  1. Propose a new sample α_{t+1} from the proposal distribution $q(\alpha_{t+1}|\theta_t)$.
  2. Calculate the acceptance ratio:

$$r = \frac{p(\alpha_{t+1}|D; I)q(\theta_t|\alpha_{t+1})}{p(\theta_t|D; I)q(\alpha_{t+1}|\theta_t)}$$

The Metropolis-Hastings algorithm

- Start with an initial guess θ_0.
- For each iteration t:
  1. Propose a new sample α_{t+1} from the proposal distribution $q(\alpha_{t+1}|\theta_t)$.
  2. Calculate the acceptance ratio:

$$r = \frac{p(\alpha_{t+1}|D;I)q(\theta_t|\alpha_{t+1})}{p(\theta_t|D;I)q(\alpha_{t+1}|\theta_t)}$$

  3. If $r \geq 1$, the we set $\alpha_{t+1} = \theta_{t+1}$. If $r < 1$, then we accept it with a probability r. This is done by sampling a random variable U from $U(0,1)$, a uniform distribution in the interval 0 to 1. If $U < r$ we set $\theta_{t+1} = \alpha_{t+1}$, otherwise we set $\theta_{t+1} = \theta_t$.

The Metropolis-Hastings algorithm

- Start with an initial guess θ_0.
- For each iteration t:
  1. Propose a new sample α_{t+1} from the proposal distribution $q(\alpha_{t+1}|\theta_t)$.
  2. Calculate the acceptance ratio:
  $$r = \frac{p(\alpha_{t+1}|D;I)q(\theta_t|\alpha_{t+1})}{p(\theta_t|D;I)q(\alpha_{t+1}|\theta_t)}$$
  3. If $r \geq 1$, the we set $\alpha_{t+1} = \theta_{t+1}$. If $r < 1$, then we accept it with a probability r. This is done by sampling a random variable U from $U(0,1)$, a uniform distribution in the interval 0 to 1. If $U < r$ we set $\theta_{t+1} = \alpha_{t+1}$, otherwise we set $\theta_{t+1} = \theta_t$.
- The MCMC method as initially proposed by Metropolis et al. in 1953, considered only symmetric proposal distributions, having the form $q(\alpha_{t+1}|\theta_t) = q(\theta_t|\alpha_{t+1})$.
- Hastings (1970) generalized the algorithm to include asymmetric proposal distributions and the generalization is commonly referred to as the Metropolis-Hastings algorithm. There are now many different versions of the algorithm.

Other MCMC algorithms

- **Gibbs sampling:** A special case of MCMC where the proposal distribution is chosen such that it is the conditional distribution of one parameter given the others.$p(\theta_i|\theta_1, ..., \theta_{i-1}, \theta_{i+1}, ..., \theta_n, D, I)$ No acceptance/rejection step is needed if the conditional distributions are sampled exactly. This is particularly useful when dealing with high-dimensional parameter spaces.
- **Hamiltonian Monte Carlo (HMC):** Uses the concept of Hamiltonian dynamics to propose new samples, allowing for more efficient exploration of the parameter space, especially in high dimensions. HMC simulates the movement of a particle through the parameter space using Hamiltonian dynamics, guided by gradients of the log-posterior. This allows HMC to propose distant states with high acceptance rates, reducing random walk behavior and improving sampling efficiency, especially for complex models.

The difference between the MCMC algorithms lies in how they propose new samples and how they handle the acceptance/rejection step. The choice of algorithm depends on the specific problem, the dimensionality of the parameter space, and the computational resources available.

Lecture and Tools

- Introduction to Statistics and Data Analysis for Physicists Gerhard Bohm, Günter Zech: Chapter 3 3.4, 3.5.3, 3.5.5, Chapter 5 5.2, 5.3
- To solve numerical integrals: `https://www.wolframalpha.com/`