工程概率统计

Probability and Statistics for Engineering

第四章 蒙特卡洛方法

Chapter 4 Monte Carlo Methods

Chapter 4 Monte Carlo Methods

- 4.1 Introduction and Theoretical Basis
- 4.2 Simulation of Random Variables
- 4.3 Practical Applications

4.1 Introduction and Theoretical Basis

- **Monte Carlo methods (蒙特卡洛方法)**, or Monte Carlo experiments, are a broad class of computational algorithms that rely on repeated random sampling to obtain numerical results.
- The underlying concept is to use randomness to solve problems that might be deterministic in principle (e.g., calculating probabilities, expectations, integrations).
- The name comes from the Monte Carlo Casino in Monaco, where the primary developer of the method, Stanislaw Ulam, was inspired by his uncle's gambling habits.
- Monte Carlo methods are widely used in various fields of science, engineering, and mathematics.
- These methods have been recognized as one of the most important and influential ideas of the 20th century.

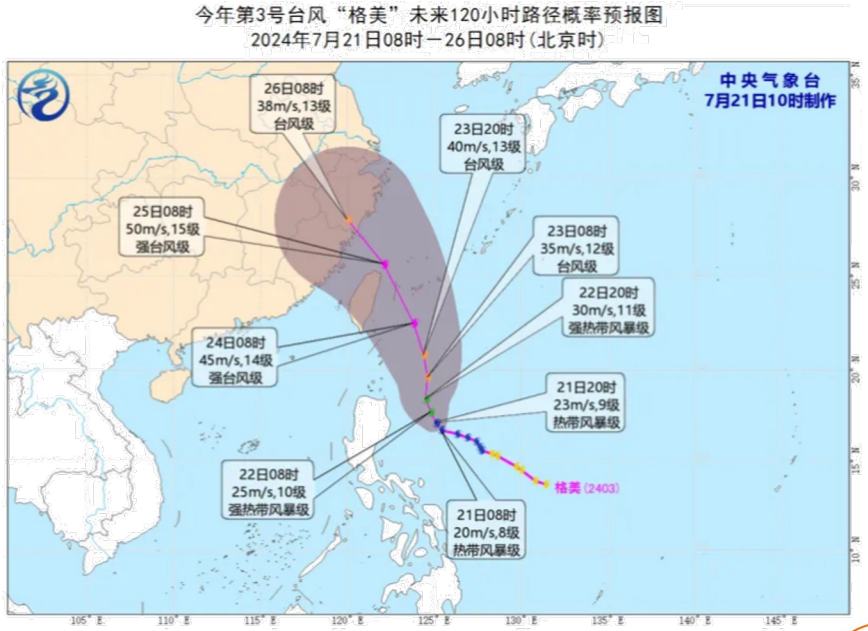
4.1 Introduction and Theoretical Basis

- Let us briefly look at a few examples, where distributions are rather complicated, and thus, Monte Carlo simulation appears simpler than any direct computation.

Example 4.1

Forecasting

- Given just a basic distribution model, it is often very difficult to make remote predictions. The prediction for tomorrow may be straightforward, but that for a week later is problematic.
- Simulation of such a process can be easily performed day by day (or even minute by minute). For every time n, we simulate X_{n+1} based on already known $X_1, X_2, \ldots, X_n$.
- Such simulations result in beautiful animated weather maps that meteorologists (气象学家) often show to us on TV news.
- They help predict future paths of storms and hurricanes as well as possibilities of flash floods (突发洪水).



4.1 Introduction and Theoretical Basis

Example 4.2

Percolation (渗流)

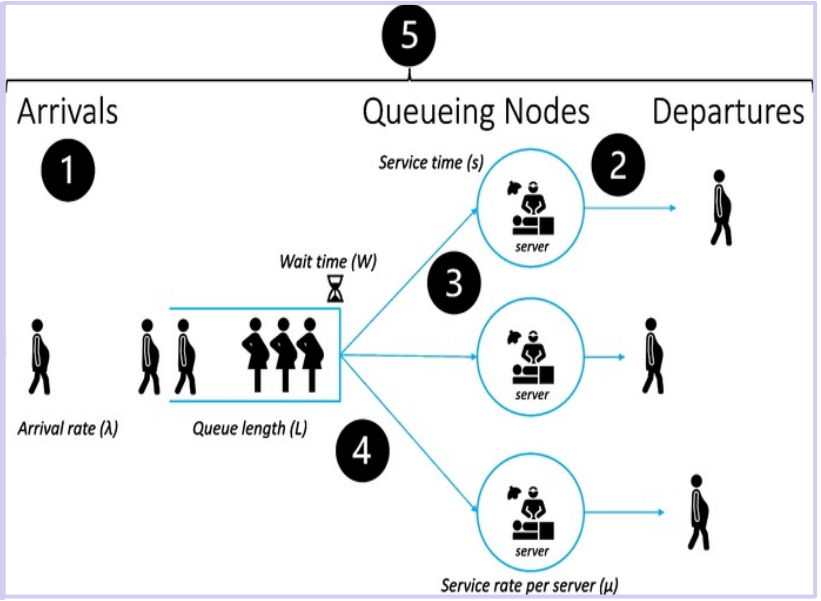
- Consider a network of nodes in which some are connected with transmission lines and a signal is sent from a node.
- Once a node k receives a signal, it sends it along its output lines with with some probability p_k.
- After some period of time, one desires to estimate the proportion of nodes that received a signal, or the probability for a certain node to receive it, etc.
- This general percolation model describes the way many phenomena may spread, e.g., a computer virus spreading between computers, rumors spreading among people, fire spreading through a forest, a disease spreading between humans.
- Simulation of such a network reduces to generating random variables $X_{kj} \sim \text{Bernoulli}(p_k)$, if $X_{kj} = 1$, then the signal transmits from node k to node j.

4.1 Introduction and Theoretical Basis

Example 4.3

Queuing System (排队系统)

- A queuing system is described by a number of random variables, involving spontaneous arrival of jobs, their random waiting time, assignment to servers, and random service time.
- The system may be complex, e.g., some jobs may not enter the system if it appears full, and intensity of the incoming traffic and number of servers on duty may change.
- When designing a queuing system or server facility, we need to evaluate its key performance characteristics.



- In all the examples, we saw how different scenarios can be computer-simulated.
- One simulation is not enough due to randomness, we generally obtain a number of realizations, from which we calculate probabilities/expectations as long-run frequencies/averages.
- The theoretical basis behind it is the **Law of Large Numbers (大数定律)**.

4.1 Introduction and Theoretical Basis

- The **law of large numbers (LLN, 大数定律)** states that the average of the results obtained from a large number of independent random samples converges to the true mean, if it exists.

Law of Large Numbers

- Let $X_1, X_2, \ldots$ be a sequence of i.i.d. random variables with $\mu \triangleq \mathrm{E}(X_i) < \infty$, $\bar{X}_n = (X_1 + X_2 + \cdots + X_n)/n$.
- Then $\bar{X}_n$ **converges in probability (依概率收敛)** to μ as $n \to \infty$:

$$\bar{X}_n \xrightarrow{\mathrm{P}} \mu, \text{as } n \to \infty.$$

- In other words, for any $\varepsilon > 0$,

$$\lim_{n\to\infty} \mathrm{P}\left(\left|\frac{1}{n}\sum_{i=1}^{n} X_i - \mu\right| > \varepsilon\right) = 0.$$

- This is actually the **weak law of large numbers (弱大数定律)**.
- There is also a strong law of large numbers, which is beyond the scope of this course.

4.1 Introduction and Theoretical Basis

- Note that $\bar{X}_n$ is a random variable, while μ is a constant. **Question:** what's the distribution of $\bar{X}_n$?

Answer: By the CLT, $\bar{X}_n$ approximately follows a normal distribution when n is large.

The mean/variance of the normal distribution can be obtained by calculating the mean/variance of $\bar{X}_n$:

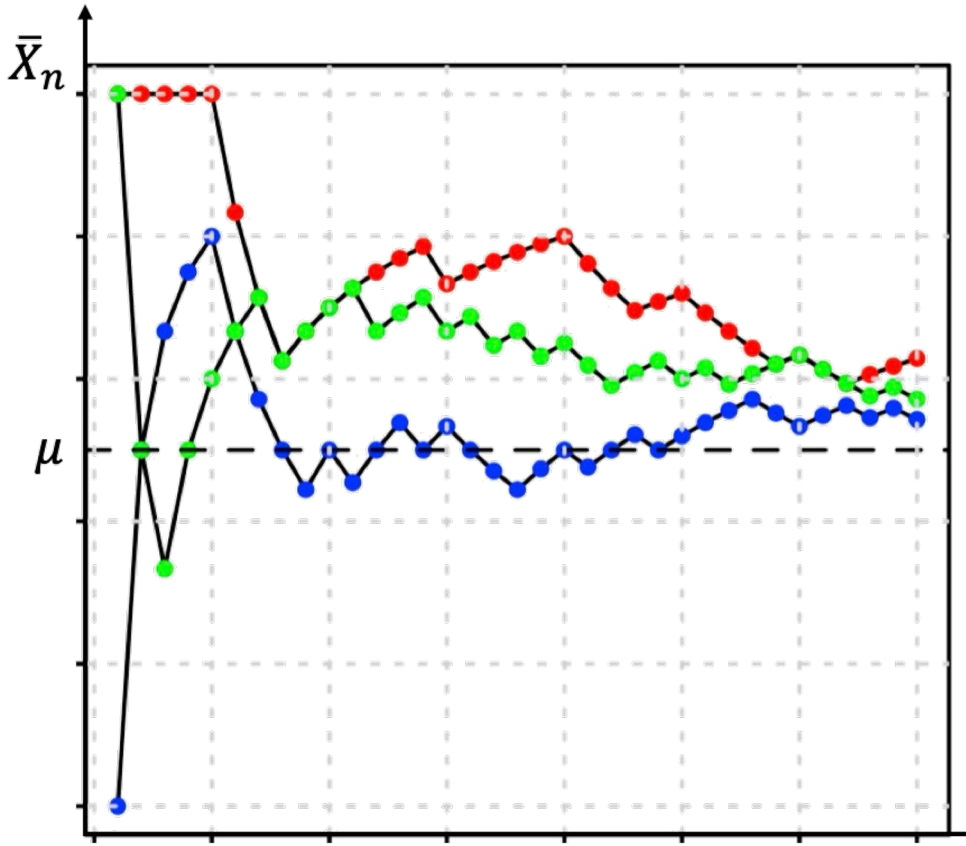
$$\mathrm{E}(\bar{X}_n) = \frac{\mathrm{E}(X_1) + \cdots + \mathrm{E}(X_n)}{n} = \mu,$$

$\sigma^2 \triangleq \mathrm{Var}(X_i)$

$$\mathrm{Var}(\bar{X}_n) = \frac{\mathrm{Var}(X_1) + \cdots + \mathrm{Var}(X_n)}{n^2} = \frac{\sigma^2}{n}.$$

$$\Longrightarrow \bar{X}_n \overset{\text{approx.}}{\sim} \mathrm{N}\left(\mu, \frac{\sigma^2}{n}\right), \text{ as } n \to \infty, \text{ the variance of } \bar{X}_n$$

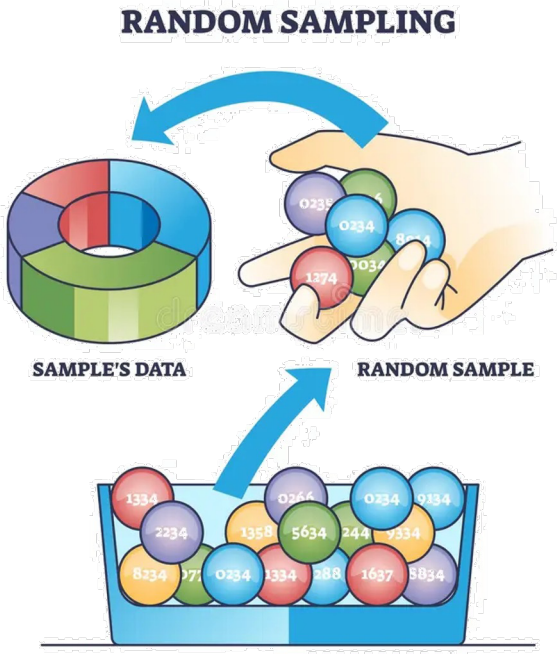
goes to zero so that $\bar{X}_n$ converge in probability to μ.



4.1 Introduction and Theoretical Basis

- A special case of the LLN is when $X_1, X_2, \ldots \overset{\text{i.i.d.}}{\sim} \text{Bernoulli}(p)$, i.e., $\text{P}(X_i = 1) = p, \text{P}(X_i = 0) = 1 - p$.
  - $\bar{X}_n = (X_1 + X_2 + \cdots + X_n)/n$ is the frequency (频率) of success, $p = \text{E}(X_i)$ is the probability of success.
  - Then the LLN states that $\bar{X}_n \xrightarrow{\text{P}} p$ as $n \to \infty$, suggesting that probability is the long-run frequency which can be approximated by calculating the frequency from repeated Bernoulli trials.

- Therefore, the LLN provides the theoretical basis for the Monte Carlo methods, which essentially calculate probabilities / expectations as long-run frequencies/averages.
- The most important step of Monte Carlo methods is to generate numbers for random variables from given distributions.
- In the next section, we will talk about random number generation.



Chapter 4 Monte Carlo Methods

- 4.1 Introduction and Theoretical Basis
- 4.2 Simulation of Random Variables
- 4.3 Practical Applications

4.2 Simulation of Random Variables

- Generating random numbers is not an easy task and it is difficult to verify "true randomness".
- Typically, a pseudo-random number generator (PRNG, 伪随机数生成器) is utilized.
  - It is an algorithm for generating a sequence of numbers whose properties approximate the properties of sequences of random numbers.
  - The generated sequence is not truly random, because it is completely determined by an initial value, called the seed.
  - It is important in practice for its speed in number generation and its reproducibility.
- Typically, a random number generator delivers to us a sequence of independent uniformly distributed numbers $u_1, u_2, \dots, u_n$ on $[0, 1]$, i.e., $\mathrm{U}[0,1]$.
- Based on $u_1, u_2, \dots, u_n$, numbers from certain simple distributions can be immediately obtained.

4.2 Simulation of Random Variables

Example 4.4

- Obtain numbers from the following distributions based on a uniform distribution random number generator:
- (1) Bernoulli(p); (2) Poisson(λ).

Solution

- (1) For random variable $U \sim \mathrm{U}[0,1]$, define X as $(0 < p < 1)$

$$X = \begin{cases} 1, & \text{if } U < p \\ 0, & \text{if } U \geq p \end{cases},$$

- then it is not difficult to show that $X \sim \text{Bernoulli}(p)$.
- Therefore, for $u_1, u_2 \ldots, u_n$ from a uniform distribution random number generator, define

$$x_i = \begin{cases} 1, & \text{if } u_i < p \\ 0, & \text{if } u_i \geq p \end{cases},$$

- then $x_1, x_2, \ldots, x_n$ can be considered numbers generated from Bernoulli(p).

4.2 Simulation of Random Variables

Solution

- (2) Consider the PMF of $\text{Poisson}(\lambda)$:

$$p_k = \mathrm{P}(X = k) = e^{-\lambda}\frac{\lambda^k}{k!}, k = 0, 1, 2, \ldots.$$

- Divide the interval $(0,1)$ into subintervals:

$$I_0 = (0, p_0), I_1 = [p_0, p_0 + p_1), I_2 = [p_0 + p_1, p_0 + p_1 + p_2), \cdots.$$

- Define $x_i = k$ if $u_i \in I_k$, then $x_1, x_2, \ldots, x_n$ can be considered numbers generated from $\text{Poisson}(\lambda)$.

- The example above shows how to generate numbers from discrete distributions, how about the case for continuous distributions?

- Recall that in Section 2.4 (Page 58), we talked about a result that can be used in random number sampling, leading to the **inverse transformation sampling (逆变换采样)**:
  - If $F(x)$ is the CDF of some continuous r.v. and its inverse function $F^{-1}(x)$ exists, let $U \sim \mathrm{U}[0,1]$, then for $X = F^{-1}(U)$ we have $X \sim F(x)$.

4.2 Simulation of Random Variables

Example 4.5

- Obtain numbers from the following distributions based on a uniform distribution random number generator:
- (1) $\mathrm{Exp}(\lambda)$; (2) $\mathrm{Gamma}(r,\lambda)$ (r is a positive integer); (3) $\mathrm{N}(\mu,\sigma^2)$.

Solution

- (1) For random variable $X\sim\mathrm{Exp}(\lambda)$, its CDF is $F(x)=1-e^{-\lambda x}$ with inverse function

$$F^{-1}(u)=-\frac{1}{\lambda}\ln(1-u).$$

- Therefore, for $u_1,u_2\ldots,u_n$ from a uniform distribution random number generator, define $x_i=-\ln(1-u_i)/\lambda$, then $x_1,x_2,\ldots,x_n$ can be considered numbers generated from $\mathrm{Exp}(\lambda)$.
- (2) The PDF of $\mathrm{Gamma}(r,\lambda)$ has been given in Section 3.3 (Page 43), the corresponding CDF is the integration of the PDF, which has no closed form expression.
- It seems that the inverse transformation sampling cannot be applied to generate numbers from $\mathrm{Gamma}(r,\lambda)$?

4.2 Simulation of Random Variables

Solution

- Actually, the fact that $\mathrm{Gamma}(r,\lambda)$ is the sum of r independent $\mathrm{Exp}(\lambda)$ can be applied.
- Since in (1), we already know how to generate numbers from $\mathrm{Exp}(\lambda)$, let $x_{11}, x_{12}, \dots, x_{1r}, x_{21}, \dots, x_{2r}, \dots, x_{n1}, \dots, x_{nr}$ be rn numbers from $\mathrm{Exp}(\lambda)$, define

$$y_i = x_{i1} + x_{i2} + \cdots + x_{ir}, i = 1, \dots, n,$$

- then $y_1, y_2, \dots, y_n$ can be considered numbers generated from $\mathrm{Gamma}(r,\lambda)$.
- (3) To sample from $\mathrm{N}(\mu,\sigma^2)$, it is enough to sample from $\mathrm{N}(0,1)$. The CDF of $\mathrm{N}(0,1)$ is $\Phi(x)$, which also does not have closed form expression. What should we do?
- It turns out that similar to Example 3.6 (Page 22 of Chapter 3), converting the problem to simulation of two independent random variables make the problem easier!
- The theory is that if $U_1 \sim \mathrm{U}[0,1]$ and $U_2 \sim \mathrm{U}[0,1]$ are independent, let

$$\begin{cases} Z_1 = \sqrt{-2\ln(U_1)}\cos(2\pi U_2) \\ Z_2 = \sqrt{-2\ln(U_1)}\sin(2\pi U_2) \end{cases}$$

Proof is left as a homework problem

- then Z_1 and Z_2 is a pair of independent standard normal random variables.

4.2 Simulation of Random Variables

Solution

- Therefore, let $u_{11}, u_{12}, u_{21}, u_{22}, \dots u_{n1}, u_{n2}$ be $2n$ numbers from $\mathrm{U}[0,1]$, define

$$z_{2i-1} = \sqrt{-2\ln(u_{i1})}\cos(2\pi u_{i2}) \text{ and } z_{2i} = \sqrt{-2\ln(u_{i1})}\sin(2\pi u_{i2}), i = 1, \dots, n,$$

- then $z_1, z_2, \dots, z_{2n}$ can be considered numbers generated from $\mathrm{N}(0,1)$.
- Consequently, $y_i = \mu + \sigma z_i$ can be considered numbers generated from $\mathrm{N}(\mu, \sigma^2)$.
- See implementations in Python.

https://docs.python.org/3/library/random.html, or https://numpy.org/doc/stable/reference/random/generator.html

- Python has built-in tools for generating random numbers from common distributions, e.g., the Binomial, Poisson, Uniform, Exponential, Gamma, Gaussian distributions, etc.

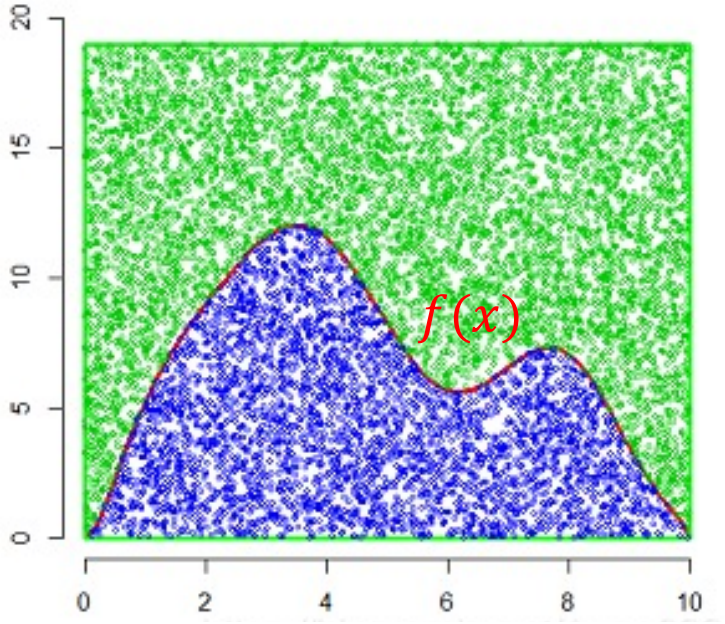
- However, there may be cases when you want to sample from some complicated distributions that are not implemented directly in computer software.
- A number of techniques have been developed to sample from complex probability distributions, e.g., the **rejection sampling (拒绝采样)** and **importance sampling (重要性采样)**.

4.2 Simulation of Random Variables

- The rejection sampling can be used given the PDF $f(x)$ of the distribution that we would like to sample from, called the **target distribution (目标分布)**.
- Since it is difficult to sample from the target distribution, rejection sampling turns to sample from a distribution with PDF $g(x)$ (called the **proposal distribution, 参考分布**) that is easy to sample from, e.g., a uniform/normal distribution.
- Then each number generated from $g(x)$ is accepted or rejected according to some criterion.
- Those accepted numbers can be considered numbers sampled from the target distribution $f(x)$.
- **Question:** What's the rationale behind this sampling technique? What proposal distribution should we choose? What criterion is used?

4.2 Simulation of Random Variables

- Rationale behind the rejection sampling technique:
  - Imagine graphing the PDF $f(x)$ onto a large rectangular board (木板) and throwing darts (飞镖) at it.
  - Assume that the darts are uniformly distributed on the board.
  - Now remove all of the darts that are outside the area under $f(x)$, then the x-positions of the remaining darts will be distributed according to $f(x)$.



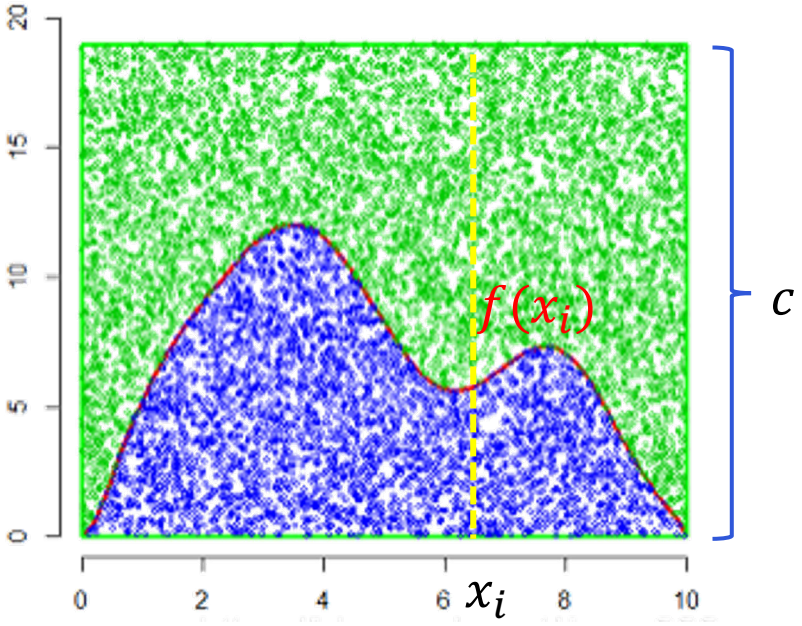
- Theoretical support:
  - Let (X, Y) have uniform distribution over the region $D = \{(x, y): 0 \le y \le f(x), a \le x \le b\}$ for some PDF $f(x)$ with support $[a, b]$, then $X \sim f(x)$.

Proof: It is essentially a problem of determining the marginal distribution of X.

$$f_X(x) = \int_{-\infty}^{\infty} f(x, y)dy = \int_0^{f(x)} 1dy = f(x).$$

4.2 Simulation of Random Variables

- In the previous rationale, the proposal distribution $g(x)$ is actually $\mathrm{U}[a,b]$ and the rejection sample is implemented as:
  - Simulate a number x_i from the proposal distribution $g(x)$ and a number y_i from $\mathrm{U}[0,c]$ (c is the height of the rectangle).
  - Then (x_i, y_i) is a point uniformly distributed on the rectangle.
  - If $y_i > f(x_i)$, then reject x_i; if $y_i \leq f(x_i)$, then x_i is a desired number from the target distribution $f(x)$.

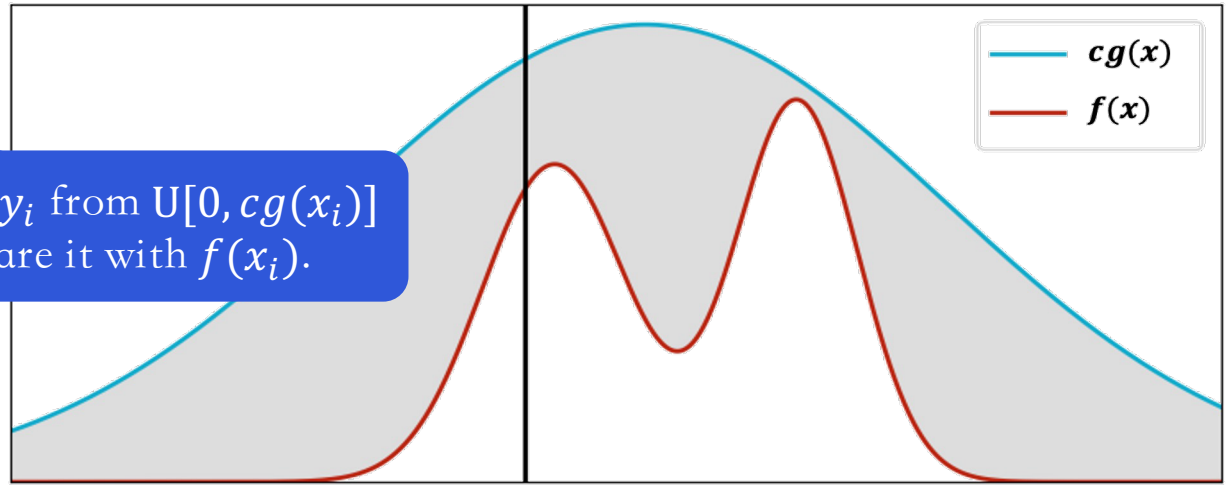


- However, the support of $f(x)$ may not be a closed interval, making a uniform proposal distribution inappropriate.
- Therefore, in the general cases, we would choose a proposal distribution $g(x)$ satisfying:
  - $g(x)$ has the same support with $f(x)$;
  - There is a constant $c > 1$ such that $f(x) \leq cg(x)$ for $\forall x$ in the support of $f(x)$.

4.2 Simulation of Random Variables

- Then the rejection sampling under the general cases is implemented as:
  - Sample x_i from $g(x)$;
  - Sample u_i from $\mathrm{U}[0,1]$, reject x_i if
$$u_i > \frac{f(x_i)}{cg(x_i)},$$
  - otherwise accept x_i.

Or to sample y_i from $\mathrm{U}[0, cg(x_i)]$ and compare it with $f(x_i)$.



- In the following, we will look at two theoretical properties of the technique.

Acceptance Probability of Rejection Sampling

- For each number x_i sampled from $g(x)$, the probability of accepting it is $1/c$. Proof:

$$\text{Acceptance Prob.} = \mathrm{P}\left(U \le \frac{f(X)}{cg(X)}\right) = \int_{-\infty}^{\infty} \mathrm{P}\left(U \le \frac{f(X)}{cg(X)} \middle| X = x\right) g(x)dx = \int_{-\infty}^{\infty} \frac{f(x)}{cg(x)} g(x)dx = \frac{1}{c}.$$

每个样本xi被接受的概率

4.2 Simulation of Random Variables

Have a higher acceptance prob.

- When implementing the rejection sampling, to make the sampling process more efficient, the constant c should be as small as possible provided it satisfies $f(x) \le cg(x)$.
- If possible, choose $c = \sup \frac{f(x)}{g(x)}$.

3. **展开联合概率：**
联合事件 “样本 $X \le t$ 且被接受” 的概率：

$$P(X \le t \text{ and accepted}) = c\int_{-\infty}^{t} P\left(U \le \frac{f(X)}{cg(X)} \mid X = x\right) g(x)dx.$$

- 条件概率 $P\left(U \le \frac{f(X)}{cg(X)} \mid X = x\right) = \frac{f(x)}{cg(x)}$。

4. **代入公式并化简：**
联合概率变为：

$$P(X \le t \text{ and accepted}) = c\int_{-\infty}^{t} \frac{f(x)}{cg(x)} g(x)dx = \int_{-\infty}^{t} f(x)dx.$$

Rejection Sampling Generates Numbers from the Target Distrib

- The accepted numbers from the rejection sampling follows the target distribution $f(x)$. Proof:
- Consider the CDF of the accepted numbers and show that it equals $F(t) = \int_{-\infty}^{t} f(x)dx$.

$$\mathrm{P}(X \le t | X \text{ is accepted}) = \frac{\mathrm{P}(X \le t, X \text{ is accepted})}{\mathrm{P}(X \text{ is accepted})} = c\mathrm{P}(X \le t, X \text{ is accepted})$$

$$= c\int_{-\infty}^{\infty} \mathrm{P}\left(X \le t, U \le \frac{f(X)}{cg(X)} \middle| X = x\right) g(x)dx = c\int_{-\infty}^{\infty} \mathrm{I}(x \le t)\mathrm{P}\left(U \le \frac{f(x)}{cg(x)}\right) g(x)dx$$

$$= c\int_{-\infty}^{\infty} \mathrm{I}(x \le t)\frac{f(x)}{cg(x)} g(x)dx = \int_{-\infty}^{t} f(x)dx = F(t).$$

被接受的样本服从f（x）分布

4.2 Simulation of Random Variables

$f^*(x)$ and $g^*(x)$ may not be PDFs as they may not integrate to 1. c_1, c_2 are called normalizing constants

- A further note:
  - We only need to know $f(x)$ and $g(x)$ up to a constant of proportionality.
  - That is, if $f(x) = c_1 f^*(x)$ and $g(x) = c_2 g^*(x)$, we can proceed with the implementation of the rejection sampling technique using $f^*(x)$ and $g^*(x)$ without having to calculate the values of c_1, c_2.

Still need to satisfy that $f^*(x) \leq cg^*(x)$ for $\forall x$ in the support of $f^*(x)$.

Example 4.6

- Please apply the rejection sampling technique to sample from

$$f^*(x) = 0.3\exp\{-(x-0.3)^2\} + 0.7\exp\{-(x-2)^2/0.3\}, x \in (-\infty, \infty).$$

Solution

- Actually, $f^*(x)$ is not a PDF as it does not integrate to 1. By comparing with the normal PDF:

$$\int_{-\infty}^{\infty} f^*(x)dx = 0.3\int_{-\infty}^{\infty} e^{-(x-0.3)^2}dx + 0.7\int_{-\infty}^{\infty} e^{-(x-2)^2/0.3}dx = 0.3\sqrt{\pi} + 0.7\sqrt{0.3\pi} \approx 1.2113.$$

- See implementations in Python.

4.2 Simulation of Random Variables

- Finally, we talk about the application of rejection sampling in Large Language Model (LLM).

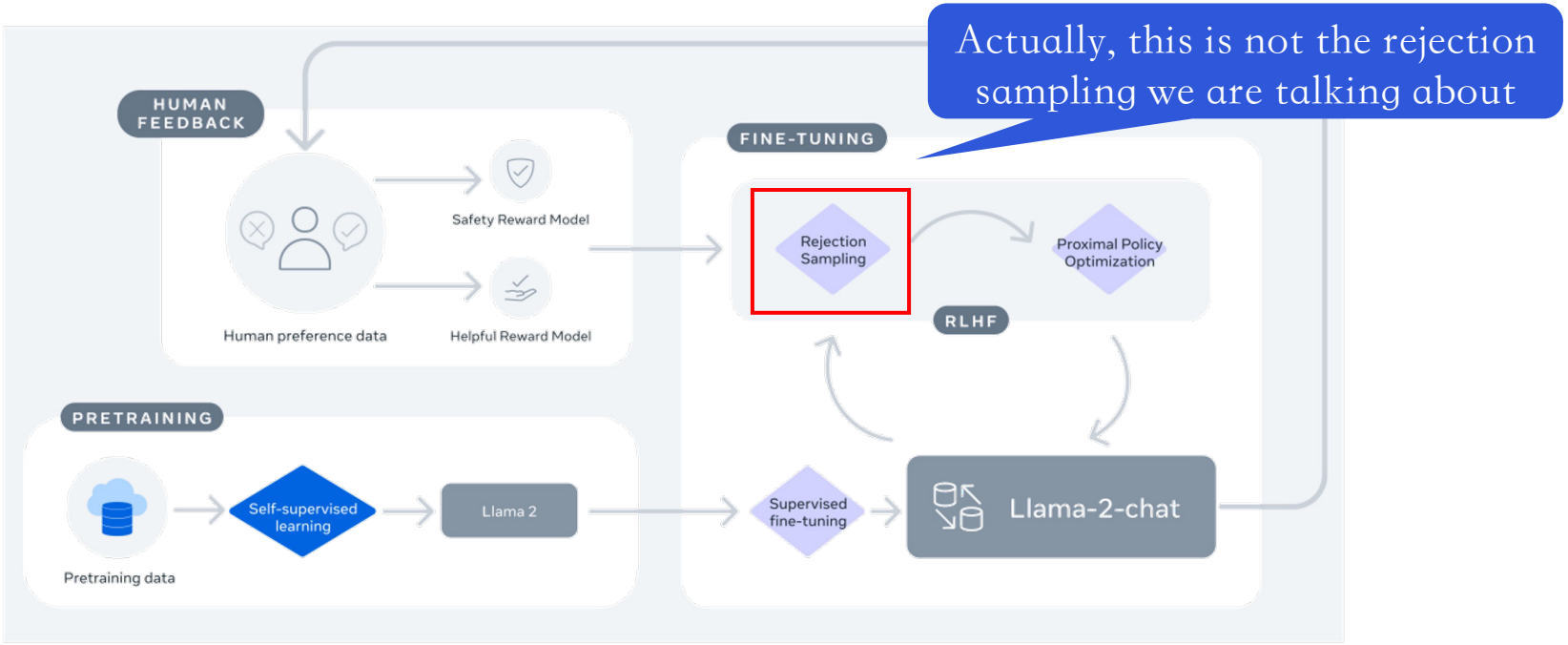


Figure 4: Training of Llama 2-Chat: This process begins with the **pretraining** of Llama 2 using publicly available online sources. Following this, we create an initial version of Llama 2-Chat through the application of **supervised fine-tuning**. Subsequently, the model is iteratively refined using Reinforcement Learning with Human Feedback (**RLHF**) methodologies, specifically through rejection sampling and Proximal Policy Optimization (PPO). Throughout the RLHF stage, the accumulation of **iterative reward modeling data** in parallel with model enhancements is crucial to ensure the reward models remain within distribution.

https://magazine.sebastianraschka.com/p/llm-training-rlhf-and-its-alternatives

(Chinese version: https://www.jiqizhixin.com/articles/2023-10-07-7)

4.2 Simulation of Random Variables

DeepMind

2023-2-3

Accelerating Large Language Model Decoding with Speculative Sampling

Charlie Chen[1], Sebastian Borgeaud[1], Geoffrey Irving[1], Jean-Baptiste Lespiau[1], Laurent Sifre[1] and John Jumper[1]

[1]DeepMind

This is the rejection sampling we are talking about

We p speculative sampling, an algorithm for accelerating transformer decoding by enabling the generatio multiple tokens from each transformer call. Our algorithm relies on the observation that the latency of p rallel scoring of short continuations, generated by a faster but less powerful draft model, is comparable to that of sampling a single token from the larger target model. This is combined with a novel modified rejection sampling scheme which preserves the distribution of the target model within hardware numerics. We benchmark speculative sampling with Chinchilla, a 70 billion parameter language model, achieving a 2–2.5× decoding speedup in a distributed setup, without compromising the sample quality or making modifications to the model itself.

Chapter 4 Monte Carlo Methods

- 4.1 Introduction and Theoretical Basis
- 4.2 Simulation of Random Variables
- 4.3 Practical Applications

4.3 Practical Applications

- Lastly, let's apply the Monte Carlo methods to solve some practical problems.

Example 4.7

- On June 12th, 2009, 5,141 families with financial difficulty in Wuhan participated in a public lottery for an affordable residential complex (经济适用住宅小区).
- As a result, 6 of the 124 lucky families had consecutive qualification certificate numbers (资格证明的编号是连续的).
- After investigation, the application materials of the 6 families were fake and their qualification was cancelled.

中国新闻网 中新网
WWW.CHINANEWS.COM

本页位置：首页 → 新闻中心 → 国内新闻

武汉经适房6连号事件黑幕:委托人"好处费"近百万

2009年07月06日 10:25 来源：《瞭望》新闻周刊 发表评论 【字体：↑大 ↓小】

"六连号"申购造假事件的背后，折射出经济适用住房政策的尴尬

文/《瞭望》新闻周刊记者

倍受关注的武汉市经济适用房"六连号"事件在十余天之后水落石出。

6月26日，由武汉市委、市政府成立，由纪检监察机关、司法机关等组成的工作组通报，这是一起由社会中介人员与有关部门工作人员相互勾结，利用经济适用房摇号进行舞弊、涉嫌经济犯罪的案件，王频等5名涉案人员已抓获到案，武汉市国土房产局副局长朱志强等5名国家工作人员因渎职、失职受到严肃处理。

4.3 Practical Applications

Example 4.7

- Coincidently, on July 29th, 2009, the lottery results of an affordable residential complex in Laohekou (老河口市, a city in Hubei) were posted online.
- People found that out of 1,138 eligible applicants, 514 were selected, in which 14 families had consecutive qualification certificate numbers.
- After multiple investigations, no illegal operations were found in this lottery.
- These two incidents inspired people's enthusiasm for calculating probabilities.
- What's the probabilities of obtaining 6 or more consecutive numbers when randomly selecting 124 out of 5,141 numbers? How about the case of obtaining 14 or more consecutive numbers when randomly selecting 514 out of 1,138 numbers?

4.3 Practical Applications

Solution

- The problem can be described as: randomly select m numbers from N consecutive numbers with equal probability, find the probability of at least k consecutive numbers among the m numbers selected, denoted by $P(N, m, k)$.
- It seems to be a classical probability problem which can be solved by permutations and combinations.
- However, the calculation is very complicated.
- By adopting a different approach, we can try to implement the Monte Carlo method using Python to compute the probabilities.
- By the Monte Carlo method using Python, we obtain:

$$P(5141, 124, 6) \approx 0.00000,$$
$$P(1138, 514, 14) \approx 0.00792.$$

The results are different if you use another random seed

- Therefore, the probability of the incident in Wuhan is much lower than that in Laohekou.

4.3 Practical Applications

Example 4.8

- A supercomputer is shared by 250 independent subscribers (用户).
- Each day, each subscriber uses the facility with probability 0.3.
- The number of tasks sent by each active user has Geometric distribution with parameter 0.15, and each task takes a $\text{Gamma}(r = 10, \lambda = 3)$ distributed computer time (in minutes).
- Tasks are processed in series.
- What is the probability that the supercomputer can finish all tasks on a randomly selected day? That is, the total requested computer time is less than 24 hours?
- Estimate this probability, attaining the margin of error ± 0.01 with probability 0.99.

4.3 Practical Applications

Solution

- First, we need to understand what are the given conditions in the problem.
- Let X be the **number of active users** on a randomly selected day, then $X \sim \text{Binomial}(250, 0.3)$.
- Given X, the **number of tasks sent to the supercomputer** is $Y = Y_1 + \cdots + Y_X$ with $Y_i \overset{\text{i.i.d.}}{\sim} \text{Geometric}(0.15)$.
- Given Y, the **total computer time** is $T = T_1 + \cdots + T_Y$ with $T_i \overset{\text{i.i.d.}}{\sim} \text{Gamma}(r, \lambda)$.
- The objective is to compute $p = \text{P}(T \leq 24 * 60) = \text{P}(T < 1440)$.
- The problem requires us to "attain the margin of error ± 0.01 with probability 0.99", we need to express its meaning mathematically:

$$\text{P}(|\hat{p}_n - p| \leq 0.01) \geq 0.99,$$

- where $\hat{p}_n$ is the frequency of finishing all tasks on a day based on n simulations.
- We would like to conduct a Monte Carlo study of size n that will guarantee an error not exceeding 0.01 with high probability 0.99.
- Therefore, before using Python to implement the Monte Carlo method, we need to determine the value of n that satisfies the requirement above.

4.3 Practical Applications

Solution

- By the CLT (see Page 6), we have

$$\hat{p}_n \overset{\text{approx.}}{\sim} \mathrm{N}\left(p, \frac{p(1-p)}{n}\right) \Rightarrow \frac{\sqrt{n}(\hat{p}_n - p)}{\sqrt{p(1-p)}} \overset{\text{approx.}}{\sim} \mathrm{N}(0,1).$$

$$\Rightarrow \mathrm{P}(|\hat{p}_n - p| \le 0.01) = \mathrm{P}\left(\frac{-0.01\sqrt{n}}{\sqrt{p(1-p)}} \le \frac{\sqrt{n}(\hat{p} - p)}{\sqrt{p(1-p)}} \le \frac{0.01\sqrt{n}}{\sqrt{p(1-p)}}\right) \approx 2\Phi\left(\frac{0.01\sqrt{n}}{\sqrt{p(1-p)}}\right) - 1$$

$$\Rightarrow 2\Phi\left(\frac{0.01\sqrt{n}}{\sqrt{p(1-p)}}\right) - 1 \ge 0.99 \Rightarrow \Phi\left(\frac{0.01\sqrt{n}}{\sqrt{p(1-p)}}\right) \ge 0.995$$

$$\Rightarrow \frac{0.01\sqrt{n}}{\sqrt{p(1-p)}} \ge \Phi^{-1}(0.995) \approx 2.575 \Rightarrow n \ge 257.5^2 p(1-p).$$

From the standard normal distribution table

- However, we have no idea about the value of p.
- The idea is to apply the fact that $p(1-p) \le 0.25$, so using $n \ge 257.5^2 * 0.25 \approx 16577$ would be safe.
- See implementation in Python.

4.3 Practical Applications

Solution

- Actually, the implementation can be simplified with the facts that:
- Given X, the number of tasks sent to the supercomputer $Y = Y_1 + \cdots + Y_X \sim \text{Negative Binomial}(X, 0.15)$;
- Given Y, the total computer time $T = T_1 + \cdots + T_Y \sim \text{Gamma}(rY, \lambda)$.

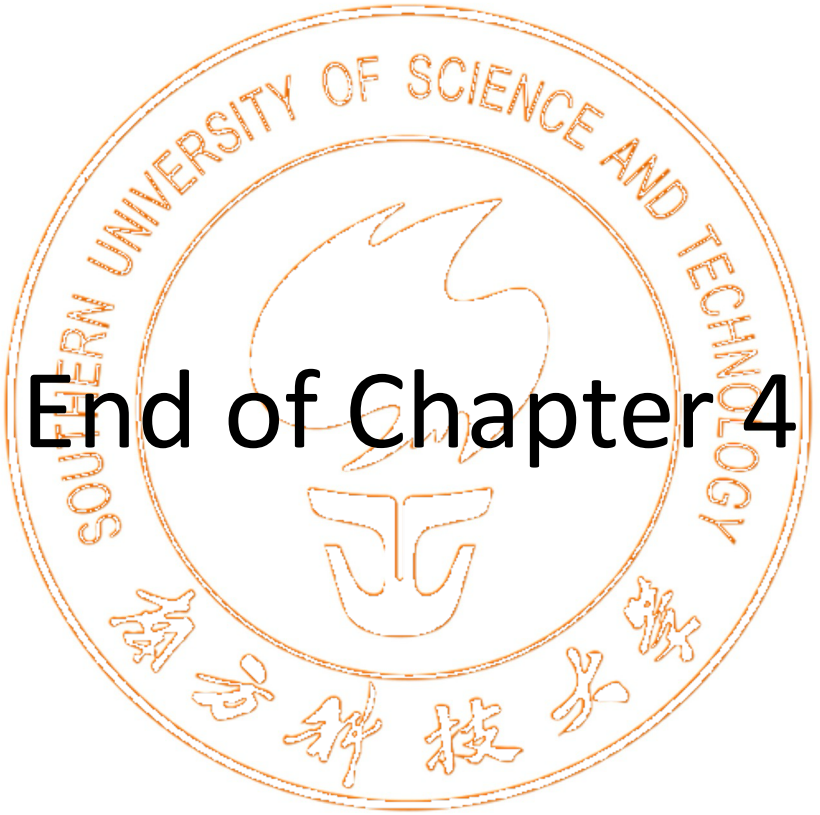
Mentioned on Page 32 in Chapter 2.

Mentioned on Page 43 in Chapter 3.

Example 4.8 (Continued)

- What is the expectation of the total requested computer time for a randomly selected day?
- What's the corresponding standard deviation?

- Monte Carlo methods are effectively used for computing probabilities, expectations, and other distribution characteristics in complex situations when calculating them by hand is difficult.
- These methods can be extended to the computation of integrals, lengths, areas, volumes, etc.



End of Chapter 4