# Theory of Monte-Carlo Analysis

EES 5891-03

Bayesian Statistical Methods

Jonathan Gilligan

Class #11: Thursday, September 29 2022

# Announcement

# Announcement

- No class Tuesday
  - I will schedule a makeup
- I will post homework due Thursday (not Tuesday) from Chapter 8
- For next Thursday, read Sections 9.4–9.6

# Monte-Carlo Analysis

# The Problem

- Bayes's Theorem $$ \begin{align} P(\beta | Y, X) &= \frac{P(Y | \beta, X) P(\beta)}{P(Y | X)} \\ &= \frac{P(Y | \beta, X) P(\beta)}{ \int P(Y | \beta, X) P(\beta) d\beta} \end{align} $$
  - $\beta$ = the set of parameters for model (a vector)
  - $Y$ = all the observed values of the outcome variable (a vector)
  - $X$ = all the observed values of the predictor variables (an array)
  - $P(Y | \beta, X)$ = *likelihood* for *Y* (e.g., $Y \sim \text{Normal}(\beta x, \sigma)$)
  - $P(\beta)$ = *prior* for $\beta$
  - $P(Y | X)$ = *evidence*
- All of these terms are super-easy to calculate except the *evidence* term: $$\int P(Y | \beta, X) P(\beta) d\beta$$

# Calculating the *Evidence*

- For a few simple cases, you can solve the integral analytically
- For most cases, there is not a simple solution
- Numerical integration:
  - Approximate the integral
  - Grid sampling:
    - We looked at this in Chapter 2–3.
    - Works for 1 and 2 dimensional problems, if the prior is close to zero for most values of \(\beta\).
    - For many dimensions (many parameters), it becomes computationally crazy.
      - 10 parameters, grid with 100 points along each dimension: $100^{10} = 10^{20}$ points. If it takes one microsecond to calculate one grid point, then it would take more than 3,000,000 years to calculate the whole grid.
  - Monte-Carlo sampling:
    - Nuclear bomb research, 1940s: calculating diffusion of neutrons in the core of a bomb.
    - Integrals were too hard for the best mathematicians
    - Stanislaw Ulam: Instead of a regular grid, pick a bunch of random numbers.
      - John von Neumann and Nicholas Metropolis made important contributions.

# Origin of Monte-Carlo Integration

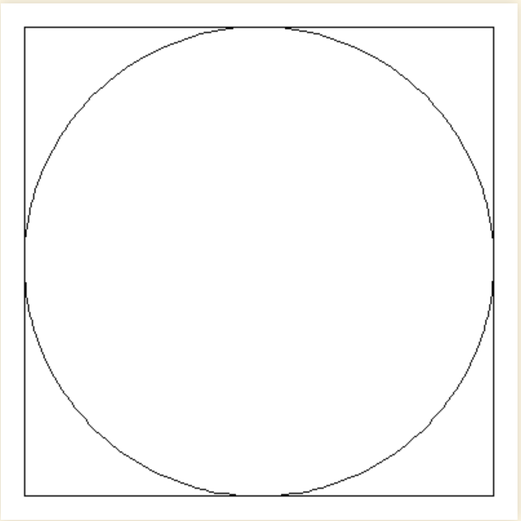
*The first were suggested by a question which occurred to me in 1946 as I was convalescing from an illness and playing solitaires. The question was what are the chances that a Canfield solitaire laid out with 52 cards will come out successfully? whether a more practical method than “abstract thinking” might not be to lay it out say one hundred times and simply observe and count the number of successful plays.*

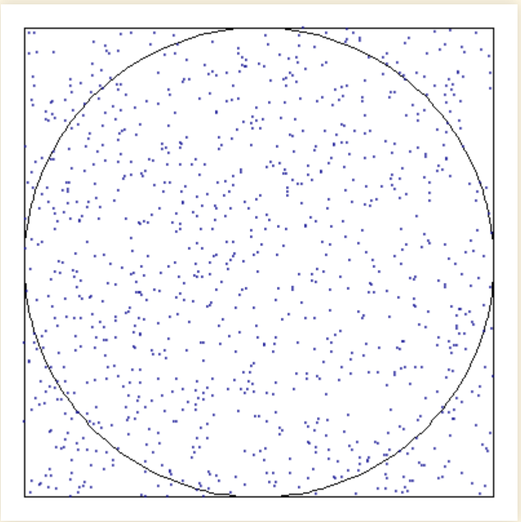
*This was already possible to envisage with the beginning of the new era of fast computers, and I immediately thought of problems of neutron diffusion and other questions of mathematical physics, and more generally how to change processes described by certain differential equations into an equivalent form interpretable as a succession of random operations.*

*— Stanislaw Ulam*

# Simple Illustration

- Estimate the area of a circle
  - Draw a square 1 inch by 1 inch
  - Inscribe a circle 1 inch in diameter
  - Throw a dart at the paper many times
  - Count the number of times the dart lands inside the circle and the number it lands anywhere inside the square $\frac{\# \text{in circle}}{\# \text{in square}} = \frac{\text{area of circle}}{\text{area of square}}$
  - 200 throws land in the square. 165 land inside the circle.
    - The ratio is 0.825.
    - The exact area of the circle is 0.785.
    - The difference is 5.0%
  - The method works just as well for the area inside any complicated shape





# Monte-Carlo for Bayesian Analysis

- Monte-Carlo:
  - If you have *n parameters* (, , , etc.):
    1. Draw *n* random numbers: each is a potential value for a *parameter*
       - These represent one point in the *n*-dimensional *parameter space* $\beta_i = (\beta_{i,1}, \beta_{i,2}, \ldots, \beta_{i,n})$
    2. Calculate *prior* and *likelihood*
    3. Repeat many times (typically a few thousand)
  - For any point $\beta_i$ in parameter space, the posterior for $\beta_i$ is $$ P(\beta_i | Y, X) = \frac{ P(Y | \beta_i, X) P(\beta_i) } { \sum_j P(Y | \beta_j, X) P(\beta_j)} $$
- The bigger *n* is, the more samples you need.
  - For high *n*, this can become very large, computationally hard.
- Can we find a smarter way to pick random numbers?
  - What does it mean to be smart about randomness?
- Markov-Chain Monte Carlo:
  - The probability distribution for the *next* random number depends on the *last* one.

# Markov-Chain Monte Carlo

# Metropolis Algorithm

1. Start at a random value for the parameter $\beta$
2. To pick the next, toss a coin.
   - Heads: your candidate is a random number greater than $\beta$.
   - Tails: your candidate is a random number less than $\beta$.
3. Calculate $q = \text{prior} \times \text{likelihood}$ at the current $\beta$, and $q'$ at the candidate point
   - if $q' > q$, move to the candidate point
   - Otherwise, pick a random number $r$ between 0 and 1.
     - If $r \le q' / q$ then move to the candidate point
     - Otherwise stay at the current point

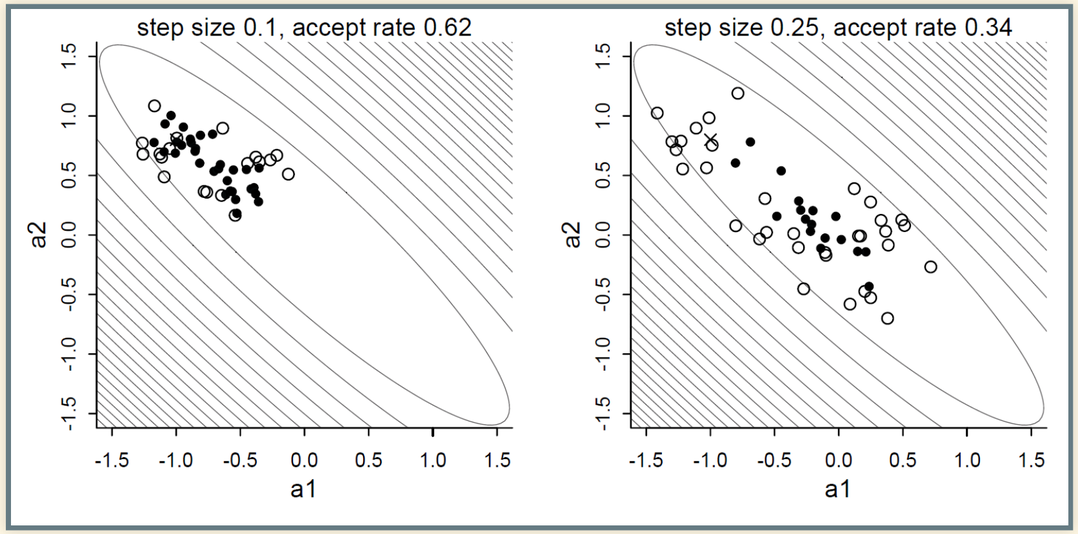
- This guarantees that you visit points where the posterior is greater more often than points where the posterior is small
  - The frequency of visiting a point is proportional to the posterior at that point
- If the posterior is relatively smooth, and only has one maximum, and is very small over most of *parameter space*, then you can estimate the integral by only looking at a small part of *parameter space*, where the posterior is significantly greater than zero.

# Gibbs Sampling

- Metropolis sampling:
  - **Symmetric:** The probability of choosing a proposal to go from $\beta_1$ to $\beta_2$ is the same a proposal to go from $\beta_2$ to $\beta_1$.
  - **Random:** You don't use any information about what you know about the posterior to choose the next point.
- Gibbs Sampling:
  - **Asymmetric** and **adaptive:** Makes smarter proposals that sample the posterior much more efficiently.
  - Cost: You can only use certain kinds of priors, called *conjugate pairs*.
  - From 1989 to 2012, Gibbs sampling was the state of the art for most Bayesian analysis.
    - R package `rjags`

# Limitations of Metropolis & Gibbs

- Metropolis and Gibbs sampling work very well for small numbers of parameters (<100 or so)
- For hundreds or thousands of parameters, they break badly.
- Even for small numbers of parameters, they break badly when parameters are highly correlated
  - (think about height and length of legs, or exercise 6M2)




- The problem is that these algorithms only look along a few directions at a time, and with high dimensions, they may not be looking in the most interesting direction, so they keep poking around in the dark, not looking at where the insteresting stuff is.

# Hamiltonian Monte Carlo

# Simple vs. Complex Monte Carlo

- Metropolis and Gibbs make simple proposals.
  - The computational cost of making a proposal is small
  - The quality of the proposals is also small
  - For simple problems, you don’t need high-quality proposals.
  - For complex problems, low-quality proposals waste time and the algorithm spins its wheels.
- Hamiltonian Monte Carlo (HMC)
  - Proposals are costly
  - But their quality is much greater
  - For complex problems, HMC finds solutions much faster.
  - For many years, HMC wasn’t very useful because you had to write custom code and there were many options to control the performance, and no one knew how to adjust the options effectively.

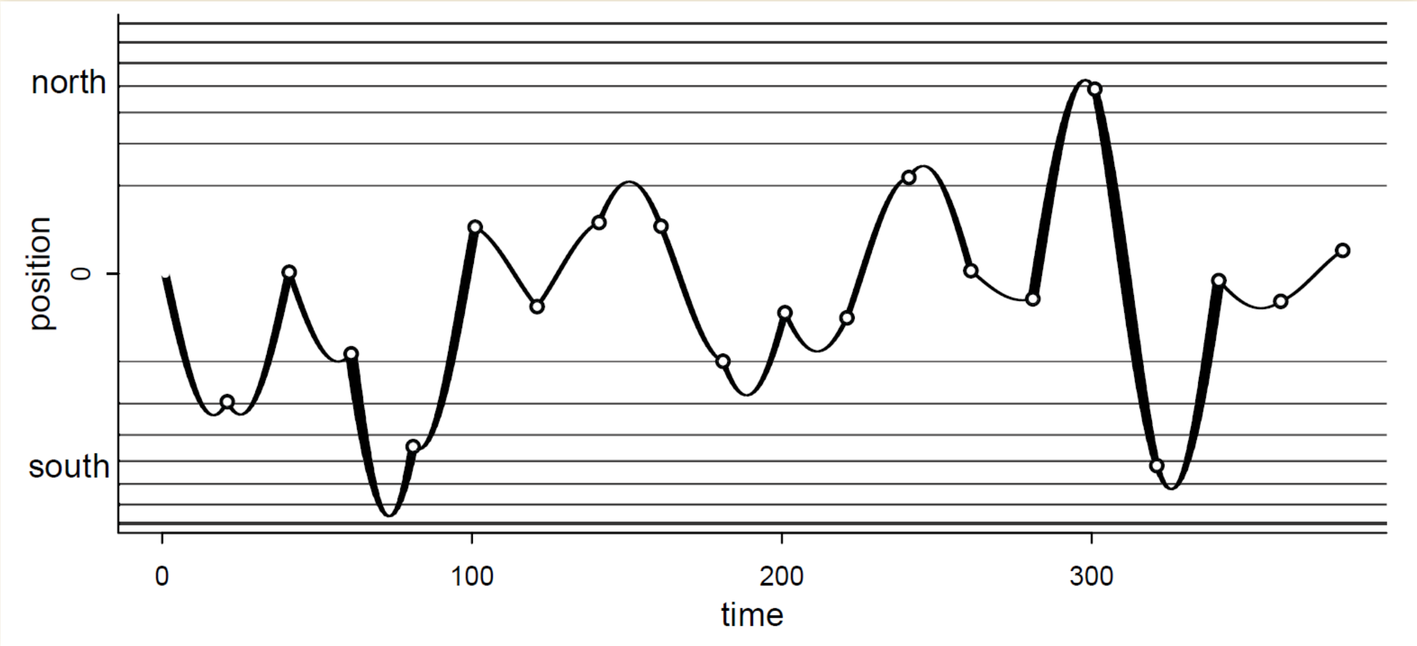
# Stan

- In 2012, the `stan` program changed everything:
  - The developers figured out how to adjust the HMC options effectively and automatically
  - `Stan` lets you write your model in a simple programming language and translates it to efficient `c++`, which is compiled and runs very fast.
  - Now R packages like `rethinking` and `brms` let you specify the model in `R` and translates it to the `stan` language, and `stan` then translates it to `c++` and turns it into a program.

# Hamiltonian Monte Carlo Carlo

- Hamiltonian Monte Carlo uses a physics simulation to do statistical calculations
  - Think of graphs of probability as hills and valleys
    - The elevation is the logarithm of the posterior probability density
  - Consider your Monte Carlo sampling point like a ball rolling on the landscape
    1. Pick a starting point and put the ball there.
    2. Flick the ball in a random direction with a random velocity
    3. Allow the ball to roll over the landscape for some amount of time
    4. Otherwise, after the time is up, wherever the ball is, that’s your next sample.

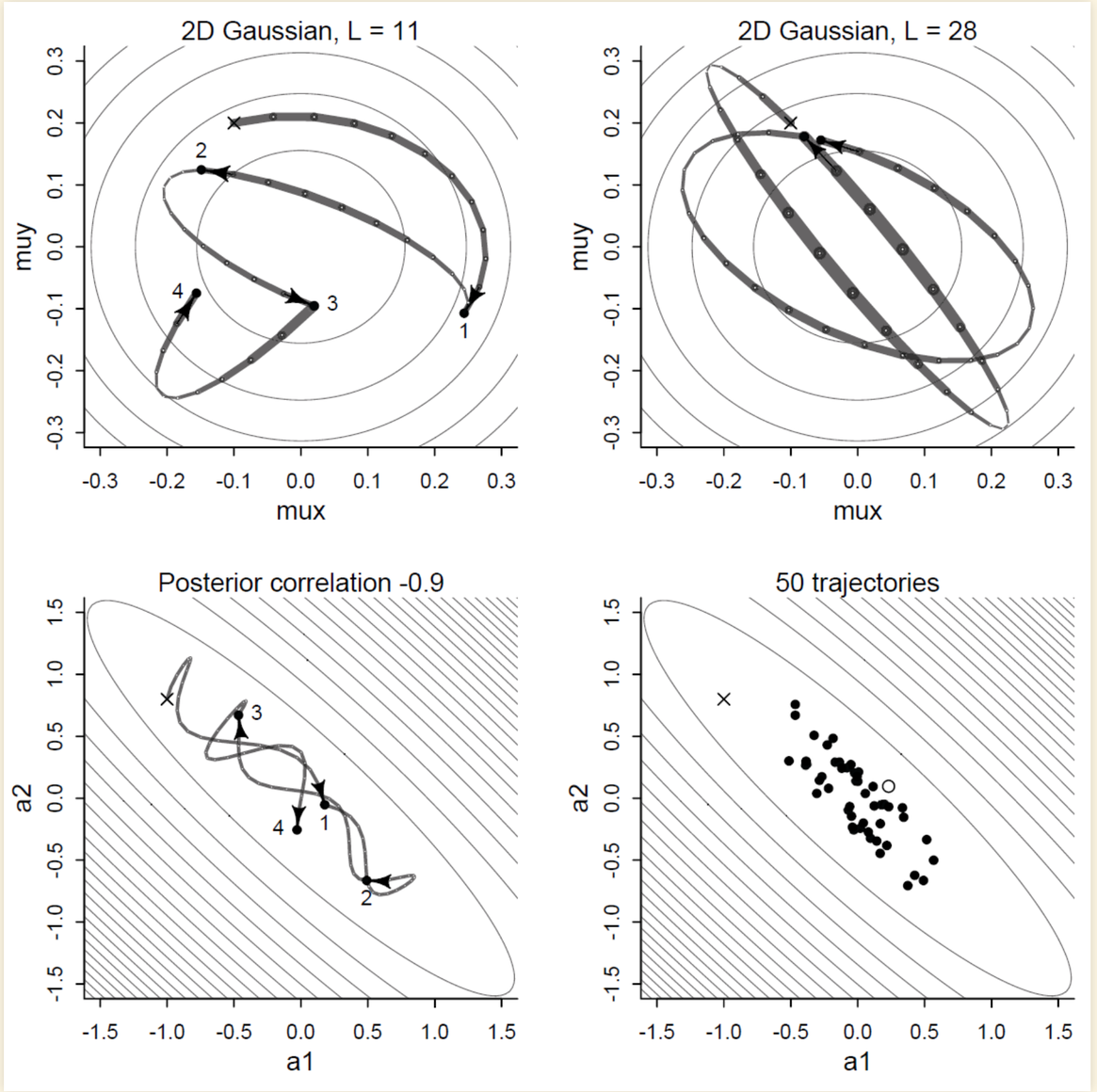
# Illustrations of Hamiltonian Monte Carlo




- Horizontal lines are elevation contours
- Position 0 is the bottom of the valleh, which goes uphill to the North and South.
- Thickness of the line is momentum (speed)
  - Ball slows down as it moves uphill
  - Ball speeds up as it moves downhill
  - Slower speed at high altitude, so it spends more time there

# Illustrations of Hamiltonian Monte Carlo

- Top left: Uncorrelated parameters, well-tuned HMC
  - Samples are far apart, uncorrelated
- Top right: Uncorrelated parameters, poorly-tuned HMC
  - Samples are close together because “ball” made U-turns
  - Stan has a “No U-Turn Sampler” (NUTS) to avoid this.
- Lower left: Highly correlated parammeters. HMC samples the space effectively, with little autocorrelation
- Lower right: Only one candidate was rejected and they effectively explore the space and quickly find the maxiumum.



# Limitations of HMC

- HMC can only work with priors that have continuous parameters and are continuously differentiable.
- Metropolis and Gibbs can solve models with discrete parameters (integrers, categories, etc.)
- You can usually find workarounds for models with discrete parameters, but this requires clever thinking
- HMC can fail and it can be very confusing to figure out why.
- But when it works well, HMC is enormously effective and much faster than Gibbs or Metropolis sampling