Introduction to probabilistic programming – part 1

Sonny Min

Department of Statistics and Actuarial Science

22 February 2024

SIMON FRASER UNIVERSITY

Outline

- Probabilistic programming
- Introduction to Bayesian inference
- Markov chain Monte Carlo (MCMC) methods
  - Metropolis-Hastings
  - Gibbs sampling
  - Hamiltonian Monte Carlo
- Summary
- References

Intro to probabilistic programming

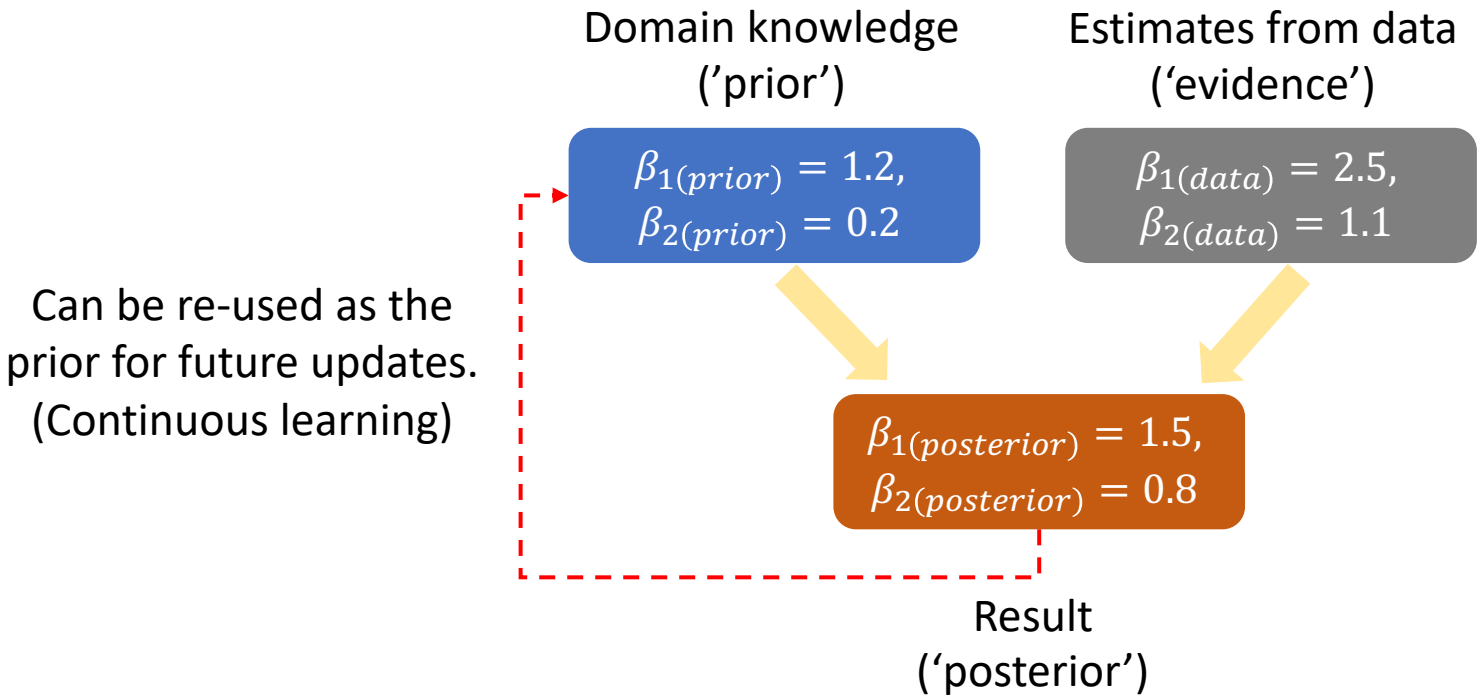
- **What is probabilistic programming?**
  - Software-driven method for specifying probabilistic models and performing inference for these models[1].
  - It makes probabilistic modelling more accessible & applicable.
    - Do not need to manually code a sampler – defining the model & parameters is enough.
  - Popular software: STAN[2], BUGS[3], JAGS[4]

[1] Hakaru – (GitHub page) "*What is probabilistic programming*"
[2] Carpenter et al. (2017) – "*Stan: a probabilistic programming language*"
[3] Lunn et al. (2000) – *"WinBUGS: a Bayesian modelling framework"*
[4] Martyn Plummer (2003) – *"JAGS: a program for analysis of Bayesian graphical models using Gibbs sampling"*

Intro to probabilistic programming

→ Probabilistic models allows incorporating domain knowledge into the model.

→ e.g. $\text{logit}(disease) = \beta_1 * Sex + \beta_2 * Alcohol + \varepsilon$



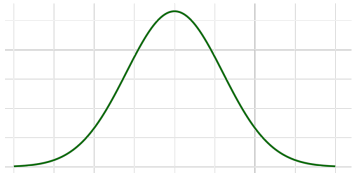
Intro to probabilistic programming

- **Probabilistic models can obtain the uncertainties of estimates**
  - PP obtains a full distribution of the parameter estimates.
  - Differentiates PP from other ML methods (DNN, tree-based methods, etc)
    - e.g. AI-predicted protein structure (AlphaFold): How certain are we about the predictions?

Parameter estimation

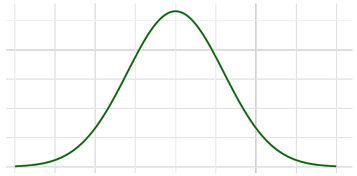
$$\text{logit}(disease) = \beta_1 * Sex + \beta_2 * Alcohol + \varepsilon$$

* Full posterior distribution of the parameters $\hat{\beta}_1$ and $\hat{\beta}_2$ are provided.



$\hat{\beta}_1$

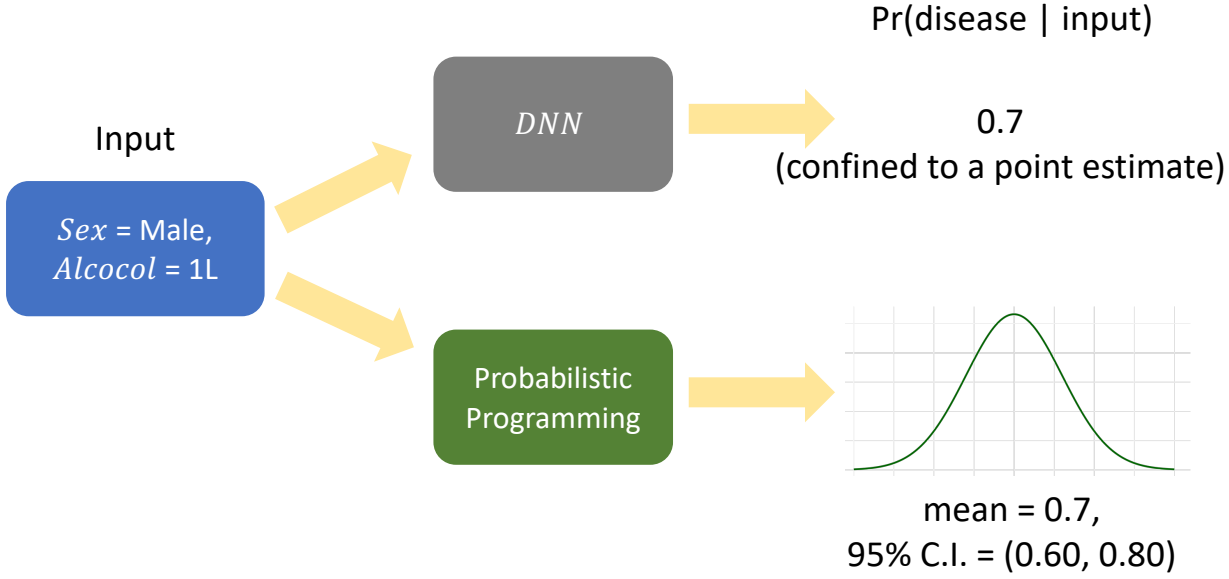
mean = 1.5
95% interval = (1.1, 1.8)



$\hat{\beta}_2$

mean = 0.8
95% interval = (0.7, 0.9)

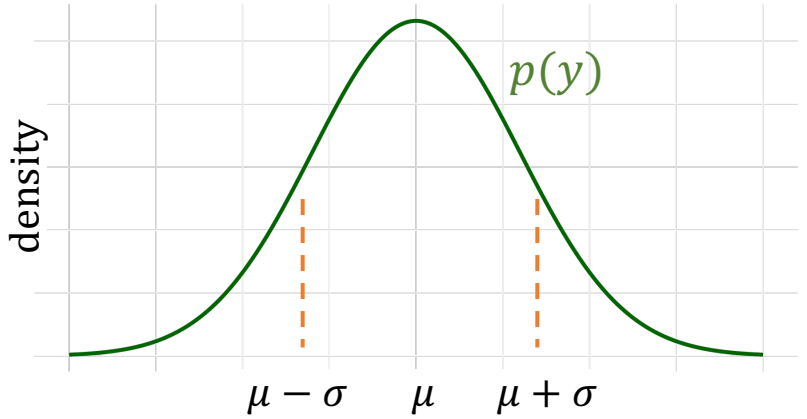
Prediction



Intro to Bayesian statistics

Random variable

- *"A variable whose values depend on the outcomes of a random event"*[5].
- e.g. Let Y = Birth weight of a newborn baby in BC.
  - Assume there is a true population mean μ, and a standard deviation σ.
  - $Y \sim Normal(\mu, \sigma^2)$
- Here, μ and σ are '**parameters**'. (Usually use 'θ' as a notation for the parameters.)
  - β_1 and β_2 in 'logit$(disease) = \beta_1 * Sex + \beta_2 * Alcohol + \varepsilon$' are also parameters.
- Probability density function (pdf)
  - pdf assigns a probability density $\in \mathbb{R}$ to each possible observation $y \in Y$.
  - $p(y) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp(-\frac{1}{2}\left(\frac{y-\mu}{\sigma}\right)^2)$



[5] Blitzstein and Hwang (2014) – "*Introduction to Probability*"

Intro to Bayesian statistics

Statistical inference

- *"The process of using data analysis to infer properties of an underlying distribution of probability"*[6]
- Infer properties of a population by hypothesis testing & parameter estimation, etc.

Frequentist vs. Bayesian

- Frequentist
  - Treat θ as a fixed value.
  - Inference of θ (i.e., obtain its estimate $\hat{\theta}$) is based solely on the data.
- Bayesian
  - Treat θ as a random variable (i.e., θ follows some distribution)
  - Infer θ by defining the distribution of θ ($p(\theta|data)$) using Bayes' rule.

$$p(A|B) = \frac{p(B|A)\,p(A)}{p(B)} \quad \rightarrow \quad p(\theta|data) = \frac{p(data|\theta)\,p(\theta)}{p(data)}$$

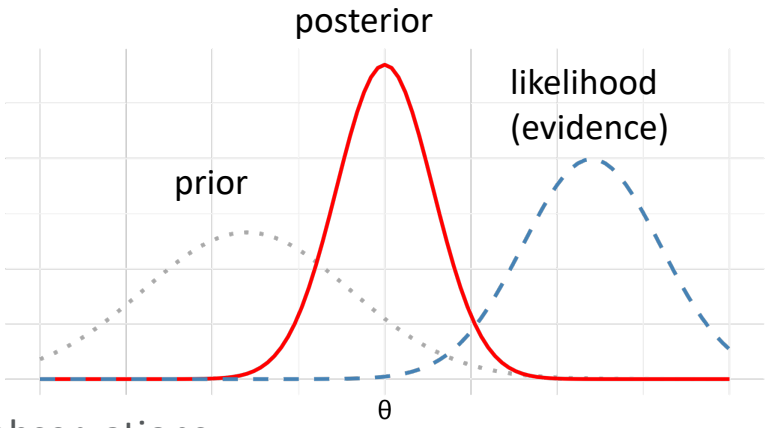
[6] Upton and Cook (2008) "Oxford Dictionary of Statistics"

Intro to Bayesian statistics

Bayesian statistics: terminology

$$p(\theta|data) = \frac{p(data|\theta)p(\theta)}{p(data)}$$

$$Posterior = \frac{Likelihood \times Prior}{Normalizing\ constant} \propto Likelihood \times prior$$



- **Posterior distribution** $p(\theta|data)$: The distribution of θ conditioned on the observations.
- **Prior distribution** $p(\theta)$: Our belief about the distribution of θ before observing the data.
- **Likelihood** $p(data|\theta)$**:** Joint probability of the observed data as a function of θ.
- **Normalizing constant** $p(data)$: A constant that ensures $p(\theta|data)$ is a probability function (i.e., sum to 1).
  - Usually difficult (often impossible) to compute directly.
  - Instead, often work with the proportional form.

Intro to Bayesian statistics

Inference by sampling: Markov Chain Monte Carlo (MCMC)

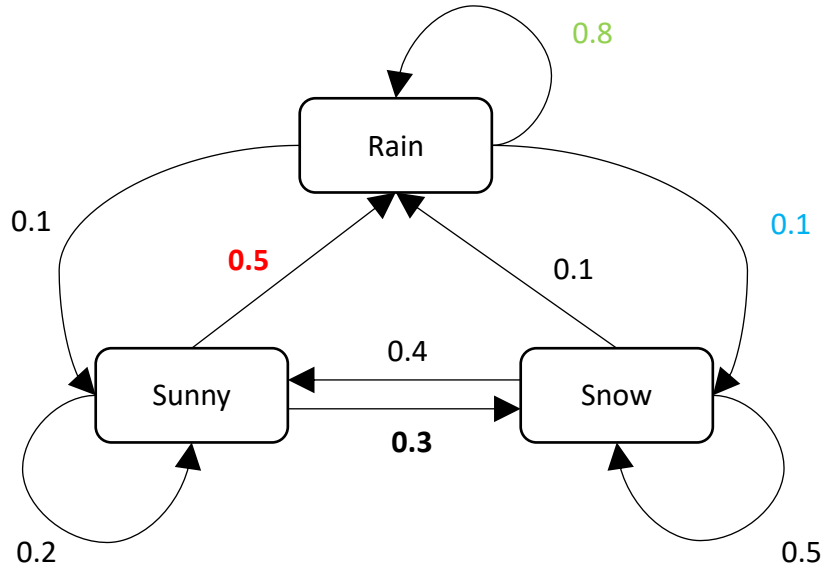
$$p(\theta|data) = \frac{p(data|\theta)p(\theta)}{p(data)} \qquad Posterior = \frac{Likelihood \times Prior}{Normalizing\ constant} \propto Likelihood \times prior$$

- The core of Probabilistic Programming.
  - Used in probabilistic programming packages (STAN, BUGS, JAGS).
- Idea: If we can (somehow) acquire samples from $p(\theta|data)$, then we can easily infer θ without having to know its full functional form.
- Does not rely on any assumptions about the data distribution.
- Asymptotically exact: As the number of samples increase, it converges to the true distribution.

Markov Chain Monte Carlo (MCMC)

Markov chain

A sequence of possible states in which the probability of each state depends only on the previous state[7].



$$\Pr(\text{Tomrrow} = rain \mid \text{Today} = sunny) = 0.5$$

$$\Pr(snow \mid rain) = 0.1$$

$$\Pr(rain \mid rain) = 0.8$$

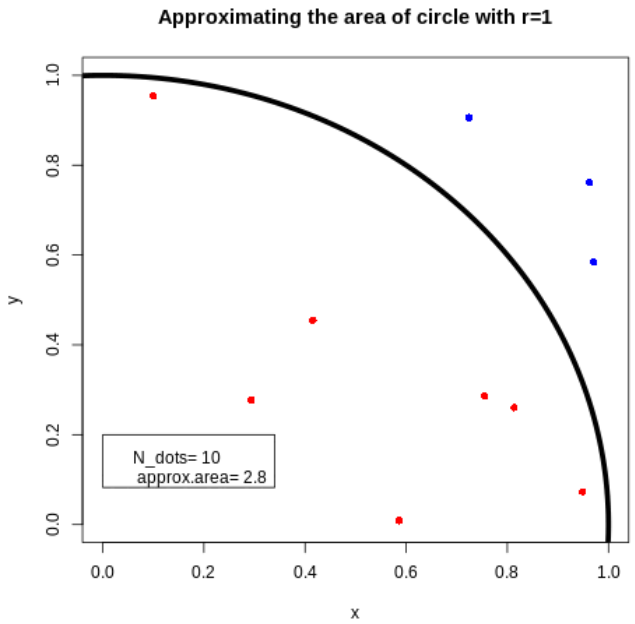
...

[7] Paul Gagniuc (2017) – "*Markov chains: from theory to implementation and experimentation*"

Markov Chain Monte Carlo (MCMC)

Monte Carlo method

- A broad class of algorithms that rely on repeated random sampling to obtain numerical results[8].
- e.g. Approximating the area of a circle with a radius = 1 unit
  - 1) Randomly draw a coordinate (x, y) where $x \in [0,1]$ and $y \in [0,1]$
  - 2) If $r = \sqrt{x^2 + y^2} \leq 1$, plot it red. Otherwise, plot it blue. (a.k.a *rejection sampling*)
  - 3) Repeat 1-2 N times.
  - $\hat{A} = \frac{\sum(red\ dots)}{N} \times 4 \approx \pi$ (as $N \to \infty$)



[8] Kroese et al. (2014) – "*Why the Monte Carlo method is so important today*"

Markov Chain Monte Carlo (MCMC)

Markov Chain Monte Carlo (MCMC)

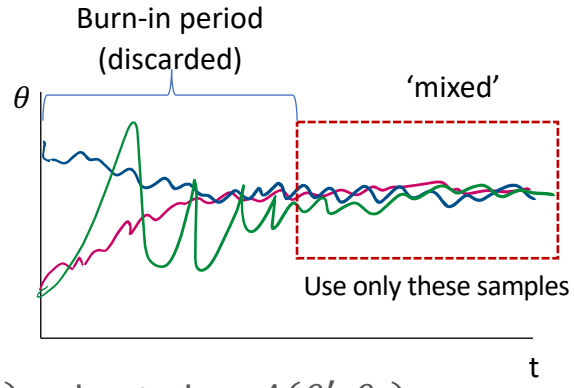
- Constructs a **Markov chain** $\theta_1, \theta_2, \dots, \theta_N$ whose stationary distribution (or the Posterior) is some distribution $P(\cdot)$.
  - A distribution $P(\cdot)$ is 'stationary' if $\theta_{t+1} \leftarrow t(\theta_t)$ where $\theta_t,\ \theta_{t+1} \sim P$.
    - $t(\cdot)$: transition distribution that moves one state to another state.
    - Future state θ_{t+1} depends only on the current state θ_t (**Markov chain**)
    - $\theta_{t+1}$is (repeatedly) randomly drawn from $t(\theta_t)$ (**Monte Carlo**)
- After obtaining large enough samples, $\hat{\theta} = \frac{1}{N}\sum_{i=1}^{N}\theta_i$

Markov Chain Monte Carlo (MCMC)

Metropolis-Hastings (MH) algorithm[9]

$$Posterior = \frac{Likelihood \times Prior}{Normalizing\ constant} \propto Likelihood \times prior$$

- Most fundamental MCMC algorithm.
- "Stationary" Target distribution $f(\cdot)$: a function that $P(\cdot) \propto f(\cdot)$, and the value of $f(\cdot)$ can be computed. (i.e. likelihood × prior)
- "transition" Proposal distribution $q(\theta'|\theta)$: an arbitrary dist'n that we can easily sample from. (e.g. Normal, Uniform, etc)
- **Intuition**: Explore the parameter space Θ via (educated) random walk provided by $q(\cdot)$, collect $\theta' \in \Theta$ that gives high $f(\theta')$
  - 1) Draw a candidate $\theta' \sim q(\theta'|\theta_t)$ (for example, $N(\theta_t, \sigma^2)$)
  - 2) Compute the **acceptance probability**: $A(\theta', \theta) = \frac{f(\theta')}{f(\theta_t)} \times \frac{q(\theta_t|\theta')}{q(\theta'|\theta_t)}$ $\in [0,1]$, and draw a constant $c \sim Unif(0,1)$
  - 3) Set $\theta_{t+1} = \begin{cases} \theta' & if\ A \geq c \\ \theta_t & if\ A < c \end{cases}$
  - 4) Repeat 1) - 3) N times. Use the accepted candidates in later sequences for $\hat{\theta}$.
- Works because $P(\cdot) \propto f(\cdot), \frac{P(\theta')}{P(\theta_t)} = \frac{f(\theta')}{f(\theta_t)}$ (See [9] for details)
- Limitation: convergence can be very slow when there are multiple parameters e. g. , $(\mu, \sigma, \gamma, \dots)$ due to low $A(\theta', \theta_t)$



[9] Wilfred Keith Hastings (1970) – "*Monte Carlo sampling methods using Markov chains and their applications*"

Markov Chain Monte Carlo (MCMC)

- **Gibbs sampler[10]**
  - Default algorithm for BUGS and JAGS.
  - Useful in multidimensional cases.
    - Pick a random starting vector $\Theta^{(0)} = \left(\mu^{(0)}, \sigma^{(0)}, \gamma^{(0)}\right)^T$
    - Draw $\mu^{(1)} \sim P(\mu \,|\sigma^{(0)}, \gamma^{(0)}, \boldsymbol{X})$
    - Draw $\sigma^{(1)} \sim P(\sigma \,|\mu^{(1)}, \gamma^{(0)}, \boldsymbol{X})$
    - Draw $\gamma^{(1)} \sim P(\gamma|\, \mu^{(1)}, \sigma^{(1)}, \boldsymbol{X})$. Now we have $\Theta^{(1)} = \left(\mu^{(1)}, \sigma^{(1)}, \gamma^{(1)}\right)^T$
    - Repeat until we get $\Theta^{(M)}$
  - Need to derive full conditional distribution of each θ: $p(\theta_j \big|\, \theta_{-j}, \boldsymbol{X})$
    - Often impossible.
    - May fail to converge if the model is too complex.

[10] Stuart and Donald Geman (1984) – *"Stochastic relaxation, Gibbs distribution and the Bayesian restoration of images"*

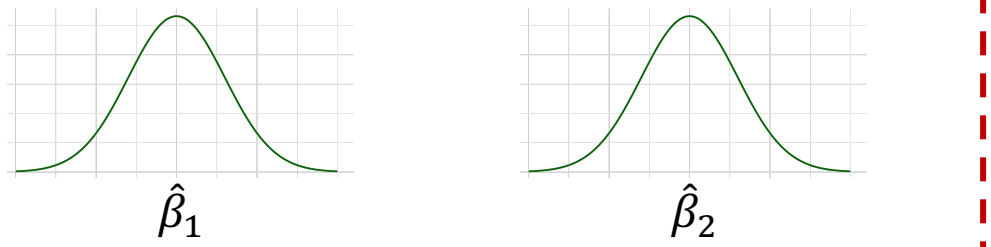
Intro to probabilistic programming

- **Probabilistic models can obtain the uncertainties of estimates**
  - PP obtains a full posterior distribution of the estimates.
  - Differentiates PP from frequentist statistics or other ML methods (DNN, tree-based methods, etc)
    - e.g. AI-predicted protein structure (AlphaFold): How certain are we about the predictions?

Parameter estimation

$\text{logit}(disease) = \beta_1 * Sex + \beta_2 * Alcohol + \varepsilon$

* Full posterior distribution of the parameters $\hat{\beta}_1$ and $\hat{\beta}_2$ are provided.



mean = 1.5
95% C.I = (1.1, 1.8)

mean = 0.8
95% C.I = (0.7, 0.9)

Prediction

Input

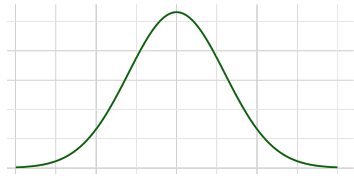
Sex = Male,
$Alcocol$ = 1L

DNN

0.7
(confined to a point estimate)

Pr(disease | input)

Probabilistic Programming



mean = 0.7,
95% C.I. = (0.60, 0.80)

Markov Chain Monte Carlo (MCMC)

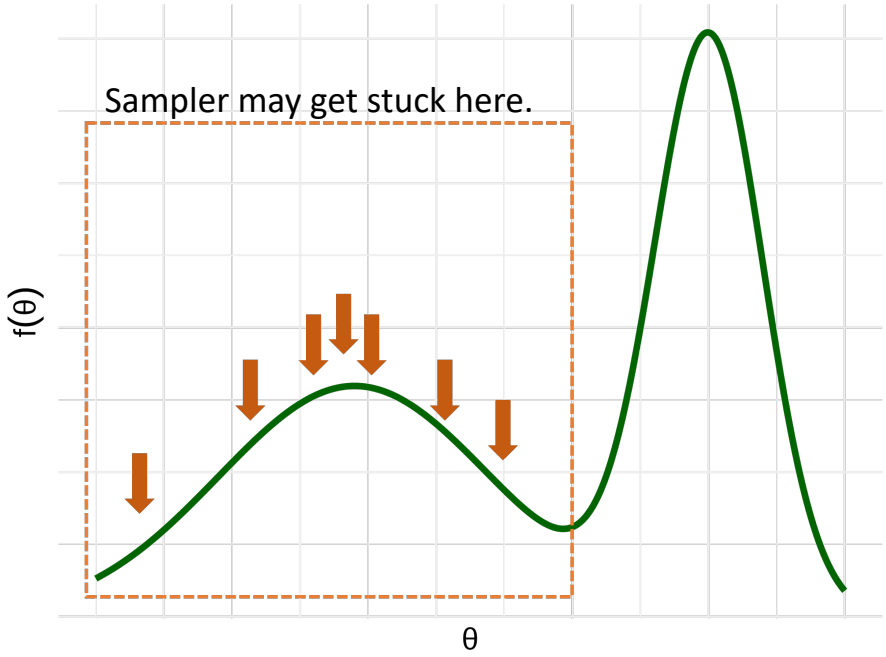
MH & Gibbs sampler: limitation

- Markov chains take small steps.
  - Parameter space is under-explored, and samples are correlated to each other.
    - More problematic in multi-modal cases.
    - The chain can get stuck in one mode, not being able to jump across multiple modes.
  - Results in a longer runtime to converge, unstable $\widehat{\Theta}$.

Keys to successful MCMC

- Good proposal & good prior.
- Make better jumps.

Example of a bimodal case



Markov Chain Monte Carlo (MCMC)

- **Hamiltonian Monte Carlo (HMC)[11]**
  - Default algorithm for STAN.
  - Makes better proposals, known to converge much faster than MH or Gibbs sampler.
  - Inspired by Hamiltonian dynamics in physics.

[11] Duane et al. (1987) – "*Hybrid Monte Carlo*"

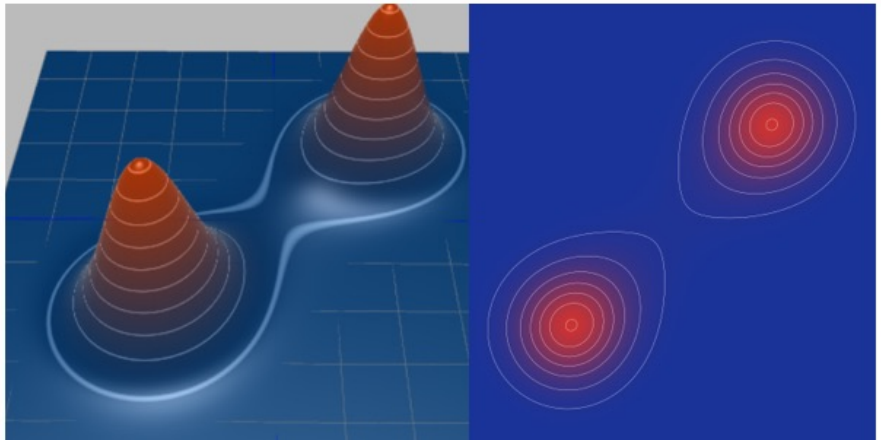
Markov Chain Monte Carlo (MCMC)

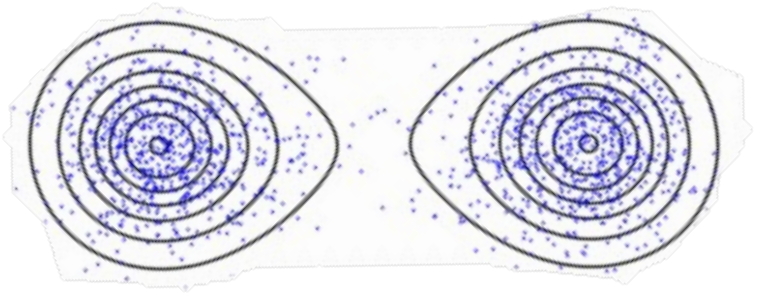
Hamiltonian Monte Carlo (HMC)

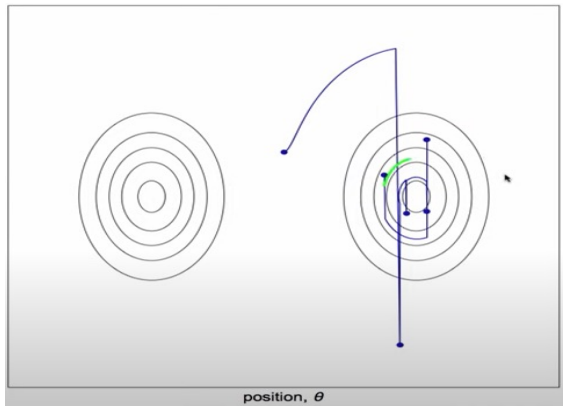
- Hamiltonian $H(\theta, p) = U(\theta) + K(p) = -\ln f(\theta) + \frac{1}{2} p^T M^{-1} p$
  - $U(\theta)$: potential energy (related to position), $K(p)$: kinetic energy (related to momentum)
  - p : 'momentum' variable to provide 'kick' (Markov chains make longer jumps)
  - M: 'mass matrix' (in Stan: diagonal estimate of the covariance computed during warmup)
  - $H(\theta, p)$ (total energy) remains approximately constant. (i.e., if the density at some θ_t is low, its kinetic energy would be high)
- Draw $p^{(0)} \sim MVN(0, M)$
- For $(i \; in \; 1{:}\, L)$, update p and θ (L: Leapfrog steps. User-defined hyperparameter):
  - $p^{(i)} \leftarrow p'^{(i-1)} + \frac{1}{2}\varepsilon \frac{d}{d\theta} \ln\left(f\left(\theta^{(i-1)}\right)\right)$
  - $\theta^{(i)} \leftarrow \theta^{(i-1)} + \varepsilon M^{-1} p^{(i)}$
  - $p'^{(i)} \leftarrow p^{(i)} + \frac{1}{2}\varepsilon \frac{d}{d\theta} \ln\left(f\left(\theta^{(i)}\right)\right)$
- $A\left(\theta^{(L)}, \theta_t\right) = \min\left\{\frac{\exp[-H(\theta^{(L)}, p'^{(L)})]}{\exp[-H(\theta_t, p^{(0)})]}, 1\right\}$, $\theta_{t+1} = \begin{cases} \theta^{(L)} & if \; A \geq c \sim Unif(0,1) \\ \theta_t & if \; A < c \sim Unif(0,1) \end{cases}$

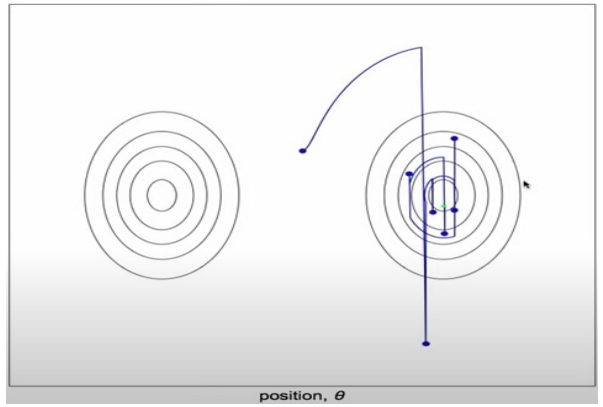
Markov Chain Monte Carlo (MCMC)

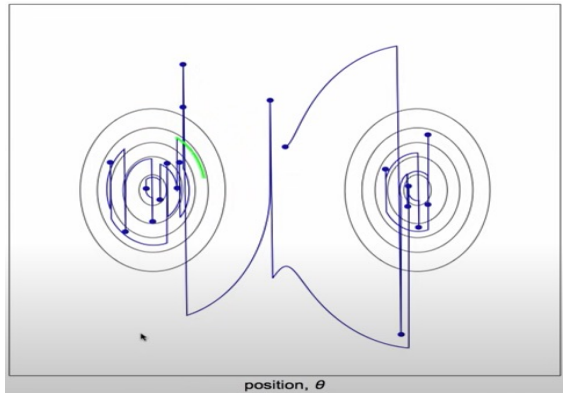
→ HMC can handle multimodal cases better [12, 13]

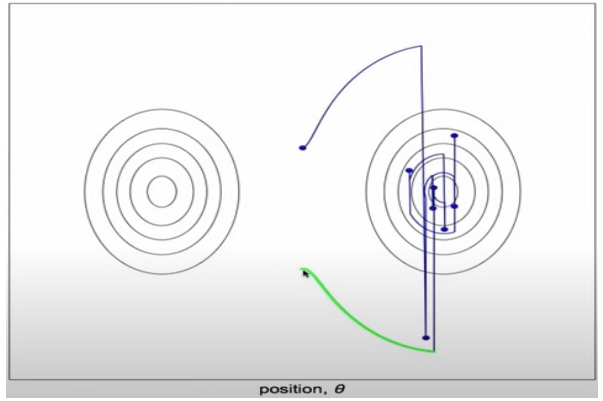












[12] Alex Rogozhnikov – (GitHub page) *"Hamiltonian Monte Carlo explained"*
[13] Ben Lambert – (YouTube video) *"The intuition behind the Hamiltonian Monte Carlo algorithm"*

Markov Chain Monte Carlo (MCMC)

- **HMC offers faster & more stable inference.**
  - Markov chains make longer jumps while maintaining a high acceptance probability.
    - Reduced correlation between samples.
  - Efficiency:
    - Fewer samples are needed for inference due to the reduced correlation.
    - For multiple parameters, instead of updating one parameter at a time, HMC moves the entire parameter space at each step.

Markov Chain Monte Carlo (MCMC)

- **STAN (Sampling Through Adaptive Neighbourhoods)**
  - Utilizes a variant of HMC (No U-turn sampler).
    - Faster & more robust inference than BUGS and JAGS.
  - Offers a more flexible modelling language.
    - Can express complex models with ease.
    - Especially useful for complex hierarchical models.
  - Provided in more software language settings.
    - STAN: R (RStan), Python (PyStan), MATLAB (MatlabStan), Julia(Stan.jl), Stata (StataStan)
    - BUGS (**B**ayesian inference **U**sing **G**ibbs **S**ampling)[3]: WinBUGS (stand-alone software), R (R2WinBUGS, RBug)
    - JAGS (**J**ust **A**nother **G**ibbs **S**ampler)[4]: JAGS (stand-alone software), R (rjags)

Summary

- **Probabilistic programming**
  - Method to automate Bayesian inference.
- **Bayesian inference**
  - Posterior = $\mathrm{p}(\theta|data) = \frac{p(data|\theta)p(\theta)}{p(data)} \propto p(data|\theta)p(\theta)$ = likelihood x prior
- **Markov Chain Monte Carlo**
  - Metropolis-Hastings & Gibbs sampler
    - Make small steps, smaller acceptance probability -> longer run time, unstable estimation.
  - Hamiltonian Monte Carlo
    - Makes longer, better jumps → faster & more stable convergence.
    - Default algorithm for STAN.

References

[1] Hakaru – (GitHub page) "*What is probabilistic programming*"

[2] Carpenter et al. (2017) – "*Stan: a probabilistic programming language*"

[3] Lunn et al. (2000) – *"WinBUGS: a Bayesian modelling framework"*

[4] Martyn Plummer (2003) – *"JAGS: a program for analysis of Bayesian graphical models using Gibbs sampling"*

[5] Blitzstein and Hwang (2014) – "*Introduction to Probability*"

[6] Upton and Cook (2008) *"Oxford Dictionary of Statistics"*

[7] Paul Gagniuc (2017) – "*Markov chains: from theory to implementation and experimentation*"

[8] Kroese et al. (2014) – "*Why the Monte Carlo method is so important today*"

[9] Wilfred Keith Hastings (1970) – "*Monte Carlo sampling methods using Markov chains and their applications*"

[10] Stuart and Donald Geman (1984) – "*Stochastic relaxation, Gibbs distribution and the Bayesian restoration of images*"

[11] Duane et al. (1987) – "*Hybrid Monte Carlo*"

[12] Alex Rogozhnikov – (GitHub page) "*Hamiltonian Monte Carlo explained*"

[13] Ben Lambert – (YouTube video) *"The intuition behind the Hamiltonian Monte Carlo algorithm"*