Introduction to probabilistic programming – part 1

Sonny Min

Department of Statistics and Actuarial Science

3 March 2022

SFU 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

Probabilistic programming

SFU

What is probabilistic programming and why?

- Software-driven method for specifying probabilistic models and performing inference for these models[1].
  - Don't need to manually code a sampler.
- You don't need a large data to begin model training.
  - Implement your domain knowledge & update the model as more evidence is acquired.
  - e.g. Predicting disease case count growth rate: Not enough data to train a good DNN model. But with probabilistic programming, you can implement your prior belief about the case count growth rate (e.g. 20%) into your model, and let the incoming data update it (say, to 30%). Here, we call the 20% the *prior*, 30% the *posterior*.
- You know the uncertainty of your estimates.
  - Probabilistic programming can give us prediction uncertainty.
  - e.g. AI predicts protein structure (AlphaFold): How certain are we about the predictions?
- Popular software: STAN, BUGS, JAGS

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

Probabilistic programming

Why STAN?

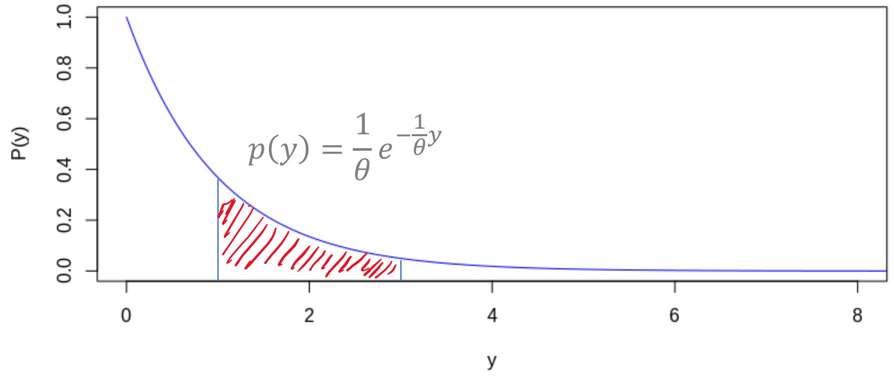
- Automatically creates Hamiltonian Monte Carlo (HMC) samplers from Bayesian model.
  - Faster than BUGS and JAGS.
- Provided in multiple software language settings.
  - STAN (**S**ampling **T**hrough **A**daptive **N**eighbourhoods)[2]:
    - R (RStan), Python (PyStan), MATLAB (MatlabStan), Julia(Stan.jl), Stata (StataStan)
  - BUGS (**B**ayesian inference **U**sing **G**ibbs **S**ampling)[3]:
    - WinBUGS (stand-alone software), R (R2WinBUGS, BRug)
  - JAGS (**J**ust **A**nother **G**ibbs **S**ampler)[4]:
    - JAGS (stand-alone software), R (rjags)

[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 Bayesian inference

Random variable

- A variable whose values depend on outcomes of a random event[5].
- e.g. Let Y = amount of time (in days) before we observe false positive from a PCR test.
  - $y \in Y$ is an observation of Y.
  - Let θ = mean days.
  - **Goal of statistical inference:** obtain an estimate $(\hat{\theta})$ for the true θ.
- A random variable is associated with a function called *'probability density function (pdf)'*
  - pdf assigns a probability density $\in \mathbb{R}$ to each possible observation $y \in Y$.
  - $Y \sim Exp(\theta), p(y) = \frac{1}{\theta} e^{-\frac{1}{\theta} y}$



Area under the curve: $p(1 \le Y \le 3) = \int_1^3 p(y)dy$

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

Intro to Bayesian inference

Frequentist vs. Bayesian

- Frequentists: *θ is a fixed (constant) value.*
  - Goal: Infer how different $\hat{\theta}$ is from a hypothesized θ_0 (e.g. $H_0: \hat{\theta} - \theta_0 = 0$)
- Bayesian: *θ is a **random variable**.* (i.e. θ has its own probability dist'n)
  - Goal: Make inference about θ by obtaining $p(\theta|data)$ using 'Bayes rule'.

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

- **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 outcomes.
- **Likelihood** $p(data|\theta)$**:** Joint probability of the observed data as a function of θ.
- **Normalizing constant** $p(data)$: A constant that reduces a function to a probability function.
  - Usually difficult (often impossible) to compute.

Intro to Bayesian inference

SFU

Bayesian inference: difficulties

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

- Bayesian inference about θ used to be a very difficult process before PP.
  - Difficult to derive the exact posterior distribution analytically.
  - High computational complexity.

Computational method: inference by sampling

- Markov Chain Monte Carlo (MCMC)
  - If we can (somehow) acquire enough samples from the posterior distribution, then we can easily obtain $\hat{\theta}$.
  - Default algorithm in STAN, BUGS, JAGS, etc.

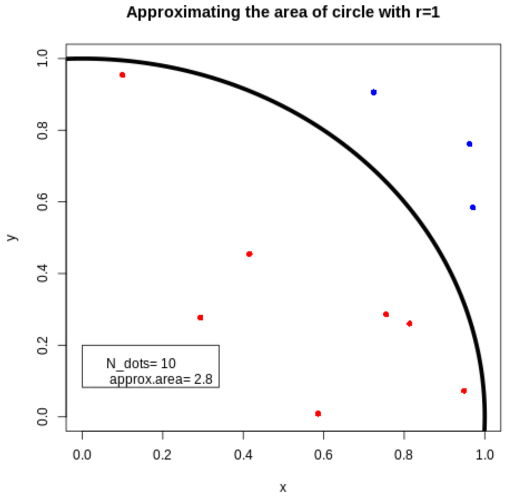
Markov Chain Monte Carlo (MCMC)

Markov chain (a.k.a Markov process)

- A sequence of possible events in which the probability of each event depends only on the state attained in the previous event[6].

Monte Carlo method

- A broad class of algorithms that rely on repeated random sampling to obtain numerical results[7].
- 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 $N \to \infty$)



[6] Paul Gagniuc (2017) – *"Markov chains: from theory to implementation and experimentation"*
[7] Kroese et al. (2014) – *"Why the Monte Carlo method is so important today"*

Markov Chain Monte Carlo (MCMC)

Markov Chain Monte Carlo (MCMC)

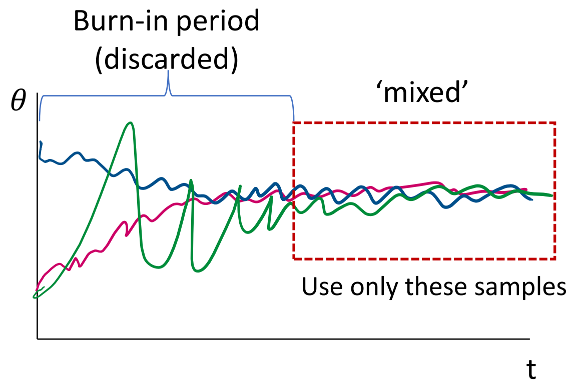
- Sampling algorithm used in probabilistic programming packages (STAN, BUGS, JAGS).
- Constructs a **Markov chain** $\theta_1, \theta_2, \dots, \theta_N$ whose *stationary distribution* is some distribution $P(\cdot)$.
  - A distribution $P(\cdot)$ is 'stationary' if

    $\theta_{t+1} \leftarrow t(\theta_t)$ where $\theta_t \sim P$ and $\theta_{t+1} \sim P$
    - $t(\cdot)$: transition distribution that moves one state to another state.
    - $\theta_{t+1}$is drawn randomly from $t(\theta_t)$ (hence **Monte Carlo**)
- $\hat{\theta} = \frac{1}{N}\sum_{i=1}^{N} \theta_i$ (With a large N)

Markov Chain Monte Carlo (MCMC)

Metropolis-Hastings algorithm[8]

- Let $P(\cdot)$: The distribution of interest we want to sample from, but hard to do so directly. (i.e. posterior)
  - 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)
  - Proposal distribution $q(\theta'|\theta)$: an arbitrary dist'n that we can easily sample from. (e.g. Normal, Uniform, etc)
- Intuition: Explore $\boldsymbol{\Theta}$ 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**: $A(\theta', \theta) = \min\{\frac{f(\theta')}{f(\theta_t)} \times \frac{q(\theta_t|\theta')}{q(\theta'|\theta_t)}, 1\}$ (i.e. $A(\theta', \theta_t) \in [0,1]$)
  - 3) Set $\theta_{t+1} = \begin{cases} \theta' & if\ A \geq c \sim Unif(0,1) \\ \theta_t & if\ A < c \sim Unif(0,1) \end{cases}$
  - 4) Repeat 1-3 N times. Use the accepted candidates in later sequences for $\hat{\theta}$.
- Works because $P(\cdot) \propto f(\cdot), \frac{P(\theta')}{P(\theta_t)} = \frac{f(\theta')}{f(\theta_t)},$
- Limitation: can be very slow for multidimensional $\Theta = (\theta_1, \theta_2, \ldots, \theta_N)$ because of low $A(\theta', \theta_t)$



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

Markov Chain Monte Carlo (MCMC)

Gibbs sampler[9]

- Default algorithm for BUGS and JAGS.
- Useful in multidimensional cases.
  - Pick a random starting vector $\Theta^{(0)} = \left(\theta_1^{(0)}, \theta_2^{(0)}, \theta_3^{(0)}\right)^T$
  - Draw $\theta_1^{(1)} \sim p(\theta_1 | \theta_2^{(0)}, \theta_3^{(0)}, \boldsymbol{X})$
  - Draw $\theta_2^{(1)} \sim p(\theta_2 | \theta_1^{(1)}, \theta_3^{(0)}, \boldsymbol{X})$
  - Draw $\theta_3^{(1)} \sim p(\theta_3 | \theta_1^{(1)}, \theta_2^{(1)}, \boldsymbol{X})$. Now we have $\Theta^{(1)} = \left(\theta_1^{(1)}, \theta_2^{(1)}, \theta_3^{(1)}\right)^T$
  - Repeat until we get $\Theta^{(M)}$
- Need to derive full conditional distribution of each θ: $\mathrm{p}\left(\theta_j \middle| \theta_{-j}, X\right)$
  - Often impossible.
- Sometimes fails to mix.

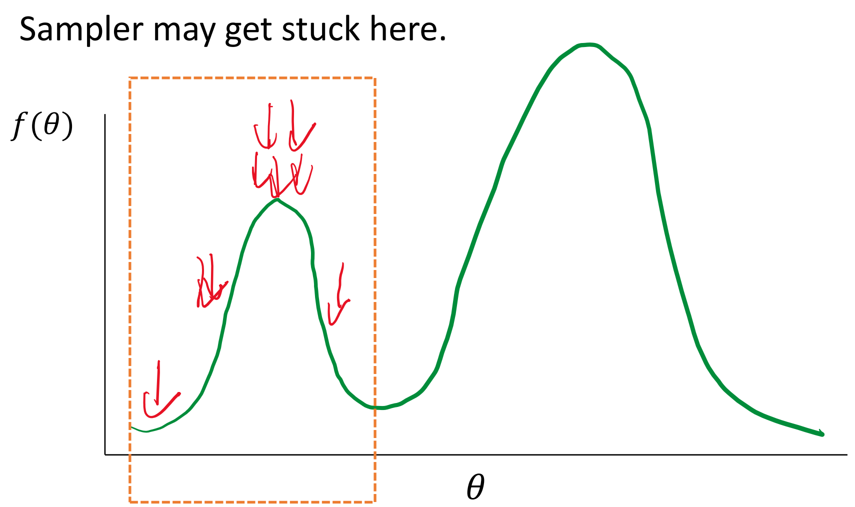
$$
\begin{aligned}
p_{cc} &\sim \text{Beta}(a_{p_{cc}}, b_{p_{cc}}) \\
S_{sen} &\sim \text{Beta}(a_{S_{sen}}, b_{S_{sen}}) \\
S_{spec} &\sim \text{Beta}(a_{S_{spec}}, b_{S_{spec}}) \\
\tau &\sim \text{Truncated Normal}(0, \infty; a_\tau, b_\tau) \\
\tau_0 &\sim \text{Truncated Normal}(0, \infty; a_{\tau_0}, b_{\tau_0}) \\
\mu &\sim \text{Truncated Normal}(0, \infty; a_\mu, b_\mu) \\
\eta &\sim \text{Truncated Normal}(0, \infty; a_\eta, b_\eta).
\end{aligned}
$$

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

Markov Chain Monte Carlo (MCMC)

MH & Gibbs sampler: limitation

- Markov chains take small steps.
  - Parameter space is under-explored.
  - More problematic in multi-modal cases.
    - The chain can get stuck in one mode instead of being able to jump across multiple modes.
  - Longer run time to mix, Unstable $\widehat{\Theta}$.



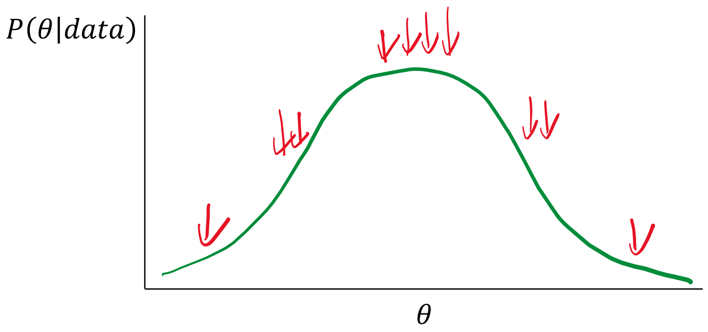
Key to successful MCMC

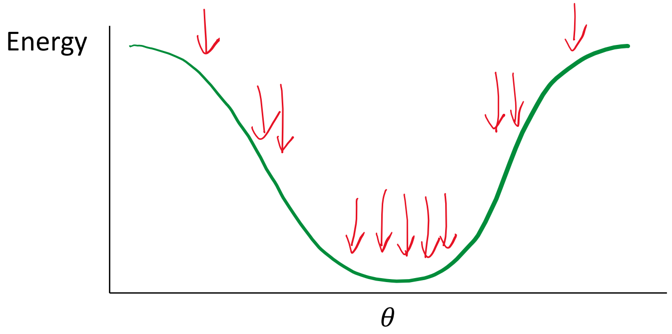
- Good proposal & good prior: there is no general rule.

Markov Chain Monte Carlo (MCMC)

Hamiltonian Monte Carlo (HMC)[10]

- Default algorithm for STAN.
- Makes better proposals. Known to mix much faster than MH or Gibbs sampler.
- Intuition from Hamiltonian dynamics in physics.





- Hamiltonian $H(\theta, p) = U(\theta) + K(p)$
  - $U(\theta)$: potential energy, $K(p)$: kinetic energy
  - Use an auxiliary momentum variable p to provide 'kick'.

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

Markov Chain Monte Carlo (MCMC)

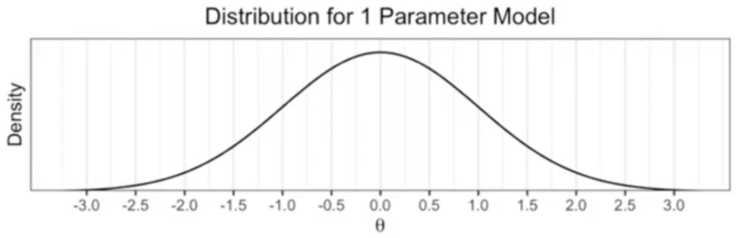
Hamiltonian Monte Carlo (HMC)

- $H(\theta, p) = U(\theta) + K(p) = -\ln f(\theta) + \frac{1}{2} p^T M^{-1} p$
- Draw $p^{(0)} \sim MVN(0, M)$
  - M^{-1}: ‘Inverse mass’. Symmetric and positive definite (Stan: diagonal estimate of the covariance computed during warmup)
- For $(i\ in\ 1{:}\ L)$:
  - $p^{(i)} \leftarrow p'^{(i-1)} + \frac{1}{2} \varepsilon \frac{d}{d\theta} \ln\left(f\left(\theta^{(i-1)}\right)\right)$
  - $\theta^{(i)} \leftarrow \theta^{(i-1)} + \varepsilon M^{-1} p^{(i)}$
  - $p'^{(i)} \leftarrow p^{(i)} + \frac{1}{2} \varepsilon \frac{d}{d\theta} \ln\left(f\left(\theta^{(i)}\right)\right)$
- $A\left(\theta^{(L)}, \theta_t\right) = \min\left\{\frac{\exp\left[-H\left(\theta^{(L)}, p'^{(L)}\right)\right]}{\exp\left[-H\left(\theta_t, p^{(0)}\right)\right]}, 1\right\}$
- $\theta_{t+1} = \begin{cases} \theta^{(L)} & if\ A \geq r \sim Unif(0,1) \\ \theta_t & if\ A < r \sim Unif(0,1) \end{cases}$

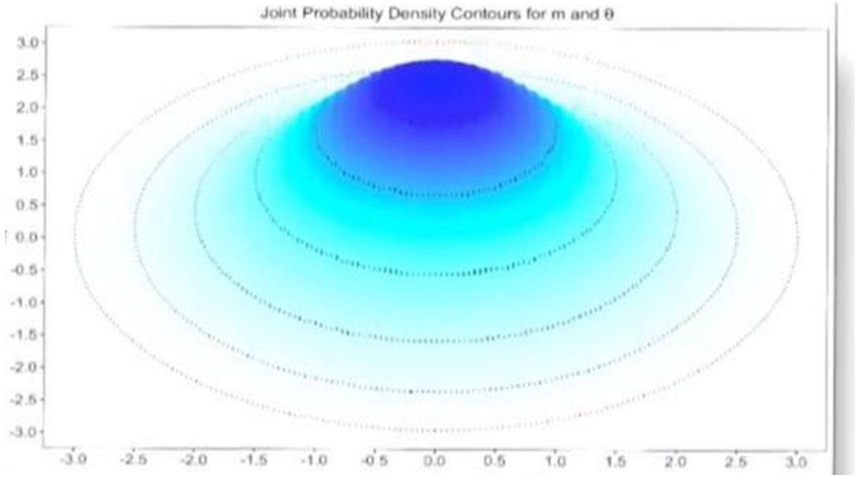
Markov Chain Monte Carlo (MCMC)

Hamiltonian Monte Carlo: example

- e.g. one-dimensional case: $\theta \sim N(0,1), \quad p \sim N(0,1)$[11]

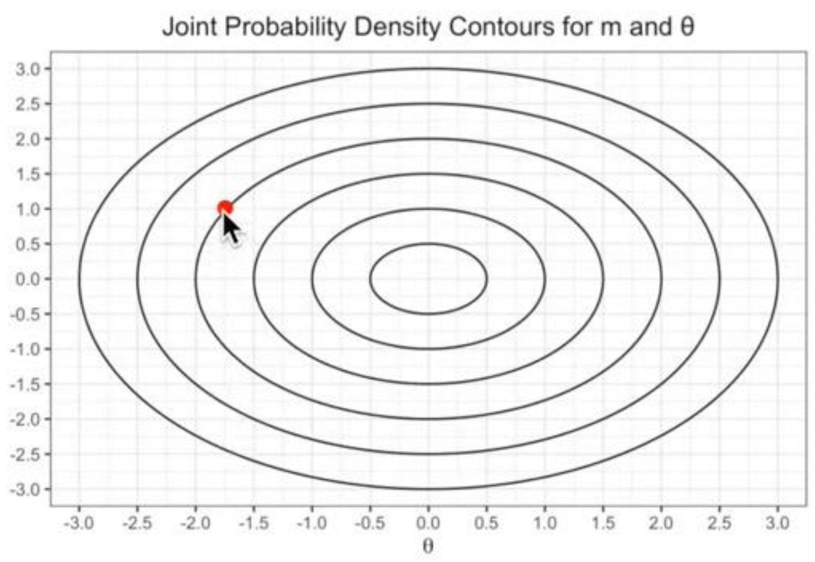


p



θ

p

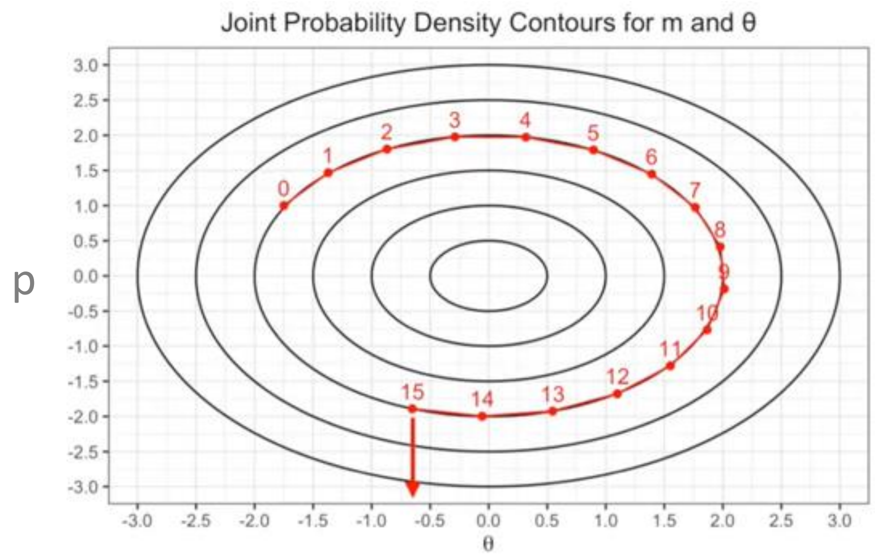


[11] Alan Malony – (YouTube video) *"Hamiltonian Monte Carlo For Dummies"*

Markov Chain Monte Carlo (MCMC)

Hamiltonian Monte Carlo: example

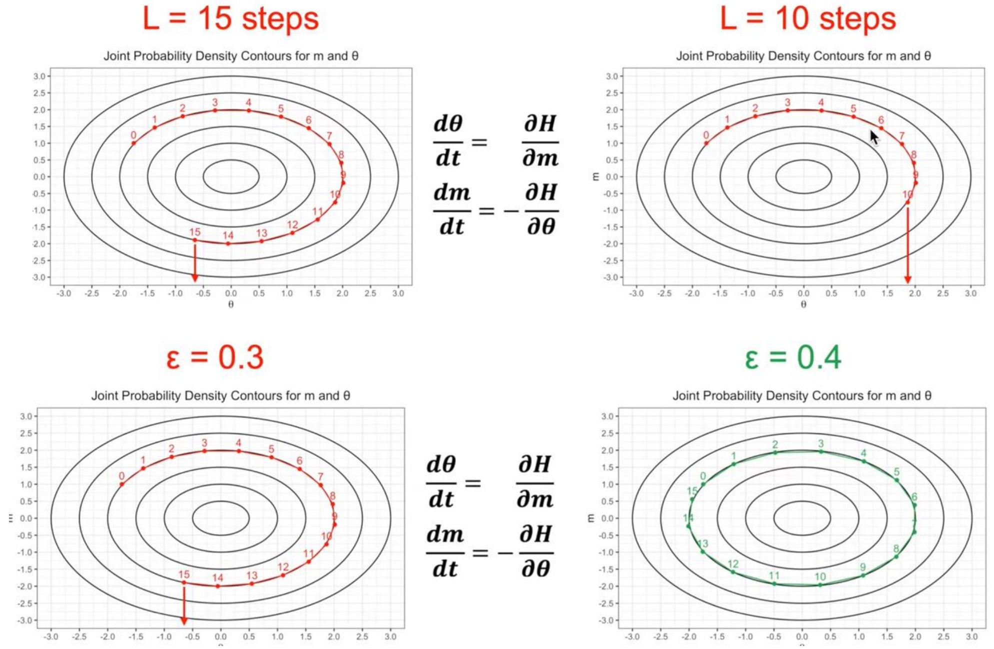
- e.g. one-dimensional case: $\theta \sim N(0,1), \;\; p \sim N(0,1)$[11]
  - 1) Set initial value for θ_t (e.g. θ_t=-1.75)
  - 2) Set initial value for p_t (e.g. p_t=1.00)
  - 3) Travel around the contour using the Hamiltonian equation.
    - Hyperparameters: L (number of steps), ε (step size)
    - Here, $L = 15$, $\varepsilon = 0.3$
  - 4) At the end of the trajectory, compute the acceptance prob. using $\theta^{(L)}, p^{(L)}, \theta_t, p_t$, and set θ_{t+1} accordingly.
  - 5) Repeat 2-4 until we get desired number of samples.



[11] Alan Malony – (YouTube video) *"Hamiltonian Monte Carlo For Dummies"*

Markov Chain Monte Carlo (MCMC)

Hamiltonian Monte Carlo: example

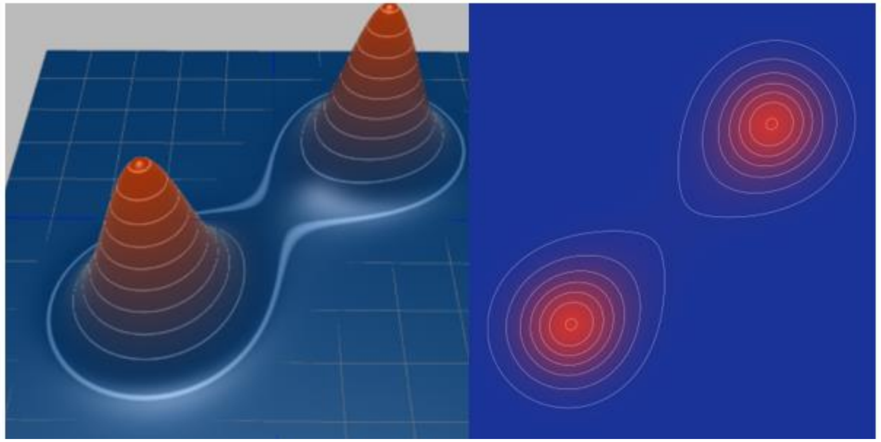


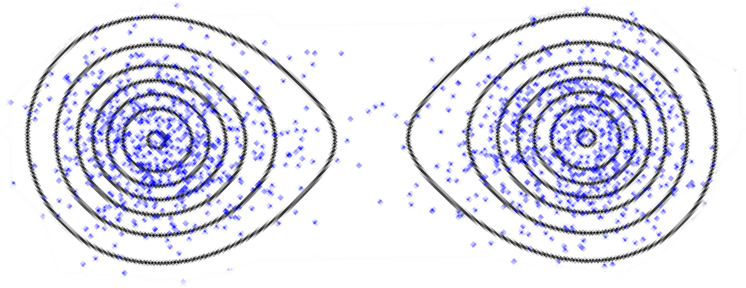
$$\frac{d\theta}{dt} = \frac{\partial H}{\partial m}$$
$$\frac{dm}{dt} = -\frac{\partial H}{\partial \theta}$$

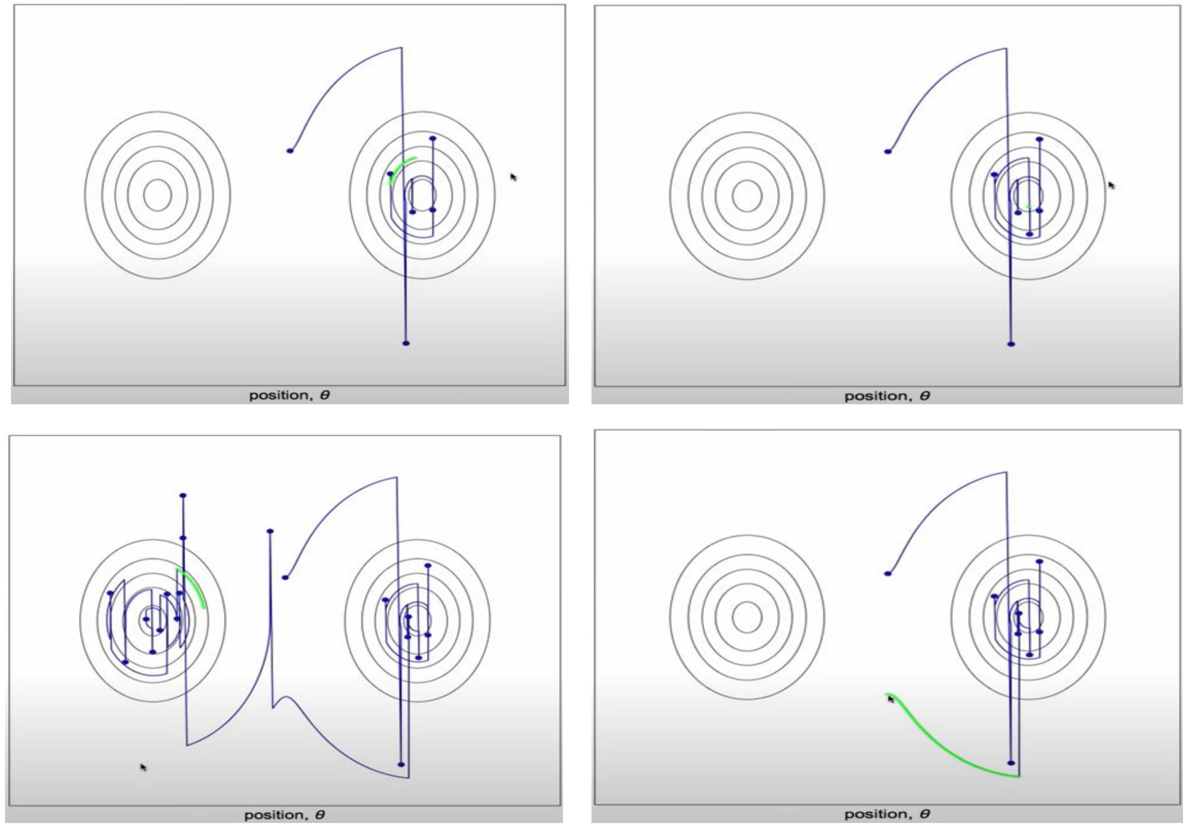
- How do we select L and ε ?
  - STAN uses 'NUTS' (No U-Turn Sampler) as a default HMC algorithm which includes L and ε optimization.

Markov Chain Monte Carlo (MCMC)

Hamiltonian Monte Carlo: bimodal case [12]







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

Summary

Probabilistic programming

- Method to automate Bayesian inference.
  - No need for manually coding a sampler.

Bayesian inference

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

Metropolis-Hastings & Gibbs sampler

- Under-representation of the posterior → longer run time, unstable estimation.

Hamiltonian Monte Carlo

- Default for RStan: creates an HMC sampler from a Bayesian model.
- Gradient-based proposal. Posterior dist’n is better represented, faster convergence.

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

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

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

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

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

[11] Alan Malony – (YouTube video) *"Hamiltonian Monte Carlo For Dummies"*

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

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