Workshop: Intro to Bayesian Learning
Lecture 5 - Introduction to Gibbs sampling, MCMC and HMC

Mattias Villani

Department of Statistics
Stockholm University

mattiasvillani.com @matvil @matvil mattiasvillani

Overview

- **Gibbs sampling**
- **The Metropolis-Hastings algorithm**
- **Hamiltonian Monte Carlo**

Monte Carlo sampling

- If $\theta^{(1)}, ..., \theta^{(m)}$ is an **iid sequence** from $p(\theta|\boldsymbol{y})$, then

$$\bar{\theta} = \frac{1}{m}\sum_{i=1}^{m}\theta^{(i)} \quad \rightarrow \quad \mathbb{E}(\theta|\boldsymbol{y})$$

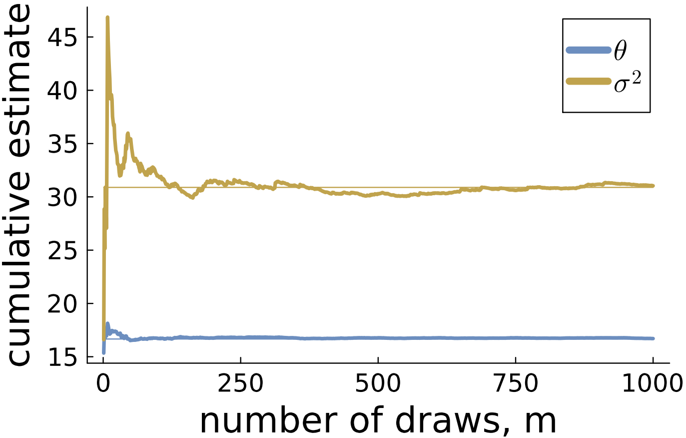
$$\bar{g}(\theta) = \frac{1}{m}\sum_{i=1}^{m}g(\theta^{(i)}) \quad \rightarrow \quad \mathbb{E}[g(\theta)|\boldsymbol{y}]$$

  for some function $g(\theta)$ of interest.

- **Central limit theorem**

$$\bar{\theta}_{1:m} \overset{\text{appr}}{\sim} N\left(\mathbb{E}(\theta|\boldsymbol{y}), \frac{\mathbb{V}(\theta|\boldsymbol{y})}{m}\right) \quad \text{for large } m$$

Monte Carlo sampling - convergence



Gibbs sampling

- **Sampling from multivariate distributions**, $p(X_1, ..., X_p)$.
- Typically a posterior distribution: $p(\theta_1, \ldots, \theta_p|\boldsymbol{y})$.
- Needed: easily sampled **full conditional posterior distributions**:
  - $p(\theta_1|\theta_2, \theta_3 ..., \theta_p, \boldsymbol{y})$
  - $p(\theta_2|\theta_1, \theta_3, ..., \theta_p, \boldsymbol{y})$
  - $\vdots$
  - $p(\theta_p|\theta_1, \theta_2, ..., \theta_{p-1}, \boldsymbol{y})$

The Gibbs sampling algorithm

Gibbs sampling

Input: initial values $\theta_2^{(0)}, \ldots, \theta_p^{(0)}$
number of posterior draws m.

for i in *1:m* **do**

$$\theta_1 \sim p\left(\theta_1 \mid \theta_2^{(i-1)}, \theta_3^{(i-1)}, \ldots, \theta_p^{(i-1)}, \mathbf{y}\right)$$

$$\theta_2 \sim p\left(\theta_2 \mid \theta_1^{(i)}, \theta_3^{(i-1)}, \ldots, \theta_p^{(i-1)}, \mathbf{y}\right)$$

$$\vdots$$

$$\theta_p \sim p\left(\theta_p \mid \theta_1^{(i)}, \theta_2^{(i)}, \ldots, \theta_{p-1}^{(i)}, \mathbf{y}\right)$$

end

Output: m autocorrelated draws for $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_p)^\top$
that converge in distribution to the joint
posterior $p(\theta_1, \ldots, \theta_p | \mathbf{y})$.

Dependent draws for Gibbs are less efficient

- $\boldsymbol{\theta}^{(1)}, ..., \boldsymbol{\theta}^{(m)}$ **converges in distribution** to posterior $p(\boldsymbol{\theta}|\boldsymbol{y})$.
- **Dependent draws** $\rightarrow$ **less efficient** than iid sampling.
- **IID samples**:

$$\mathrm{Var}(\bar{\theta}) = \frac{\sigma^2}{m}, \qquad \text{where } \sigma^2 = \mathbb{V}(\theta|\boldsymbol{y})$$

- **Autocorrelated samples**:

$$\mathrm{Var}(\bar{\theta}) = \frac{\sigma^2}{m}\left(1 + 2\sum_{k=1}^{\infty}\rho_k\right)$$

  where ρ_k is the autocorrelation at lag k.
- **Inefficiency factor**:

$$\mathrm{IF} = 1 + 2\sum_{k=1}^{\infty}\rho_k \approx 1 + 2\sum_{k=1}^{K}\rho_k$$

- **Effective sample size (ESS)**: $\frac{m}{\mathrm{IF}}$.

Gibbs sampling bivariate normal

- **Joint distribution**

$$\begin{pmatrix} \theta_1 \\ \theta_2 \end{pmatrix} \sim N_2 \left[\begin{pmatrix} \mu_1 \\ \mu_2 \end{pmatrix}, \begin{pmatrix} \sigma_1^2 & \rho\sigma_1\sigma_2 \\ \rho\sigma_1\sigma_2 & \sigma_2^2 \end{pmatrix} \right]$$

Gibbs sampling from a bivariate normal

Input: initial value $\theta_2^{(0)}$
number of posterior draws m.

for i in *1:m* **do**

$\theta_1^{(i)} \mid \theta_2 \sim \mathrm{N}\left(\mu_1 + \rho\frac{\sigma_1}{\sigma_2}(\theta_2^{(i-1)} - \mu_2), \sigma_1^2(1-\rho)^2\right)$

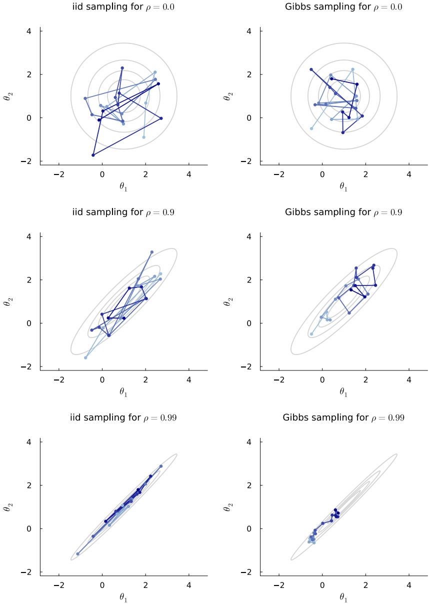
$\theta_2^{(i)} \mid \theta_1 \sim \mathrm{N}\left(\mu_2 + \rho\frac{\sigma_2}{\sigma_1}(\theta_1^{(i)} - \mu_1), \sigma_2^2(1-\rho)^2\right)$

end

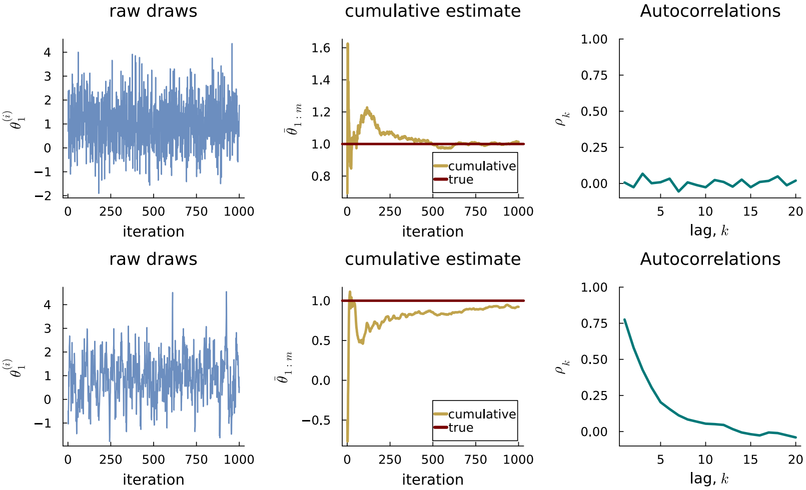
Output: m autocorrelated draws for $\boldsymbol{\theta} = (\theta_1, \theta_2)^\top$ that converge in distribution to the bivariate normal distribution $\boldsymbol{\theta} \sim \mathrm{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, where $\boldsymbol{\mu} = (\mu_1, \mu_2)^\top$ and

$$\Sigma = \begin{pmatrix} \sigma_1^2 & \rho\sigma_1\sigma_2 \\ \rho\sigma_1\sigma_2 & \sigma_2^2 \end{pmatrix}$$

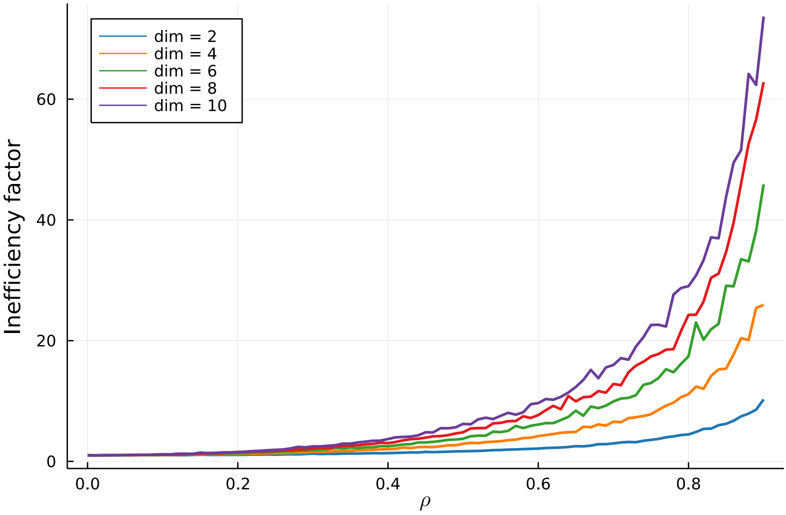
Direct sampling vs Gibbs sampling



Direct vs Gibbs sampling, bivariate normal $\rho = 0.9$



Gibbs is inefficient when parameters are correlated



Bayesian learning of ridge regularization parameter

- Cross-validation is often used to determine λ.
- Bayesian: λ is **unknown** $\Rightarrow$ **use a prior** for λ.
- $\lambda^{-1} \sim \text{Inv-}\chi^2(\omega_0, \psi_0^2)$. The user specifies ω_0 and ψ_0^2.
- Joint posterior

$$p(\boldsymbol{\beta}, \sigma^2, \lambda | \boldsymbol{y}, \boldsymbol{X})$$

- Marginal posterior λ.
- Gibbs sampling

Gibbs sampling for ridge regularization parameter

Gibbs sampling linear regression - L2 regularization prior

The posterior for the linear regression model

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \varepsilon,\ \varepsilon \sim N(\mathbf{0}, \sigma^2 I_n), \tag{11.16}$$

with hierarchical L2 regularization prior

$$\begin{aligned}\boldsymbol{\beta}|\sigma^2, \lambda &\sim N(\mathbf{0}, (\sigma^2/\lambda) I_p) \\ \sigma^2 &\sim \text{Inv}-\chi^2(\tau_0^2, \nu_0) \\ \lambda^{-1} &\sim \text{Inv}-\chi^2(\omega_0, \psi_0^2).\end{aligned}$$

can be sampled by a two-block Gibbs sampler:

$$\begin{aligned}\text{Block1}:\ \boldsymbol{\beta}|\sigma^2, \lambda, \mathbf{y} &\sim N\left(\hat{\boldsymbol{\beta}}_{L_2}, \sigma^2(\mathbf{X}^\top\mathbf{X} + \lambda I_p)^{-1}\right) \\ \sigma^2|\lambda, \mathbf{y} &\sim \text{Inv}-\chi^2(\tau_n^2, \nu_n)\end{aligned}$$

$$\text{Block2}:\ \lambda^{-1}|\boldsymbol{\beta}, \sigma^2, \mathbf{y} \sim \text{Inv}-\chi^2(\omega_n, \psi_n^2),$$

Gibbs sampling

- Gibbs sampling can be the best approach:
  - when **correlated parameters can be sampled as a block**
  - when sampling of the large blocks is fast
- Other samplers (e.g. **Metropolis-Hastings**) **within Gibbs**.
- **Data augmentation** can make Gibbs more applicable:
  - Probit regression
  - Logistic regression
  - Horseshoe-regularized regression and classification
  - Mixture models

Data augmentation - Probit regression

- **Probit regression**:

$$\Pr(y_i = 1 \mid \boldsymbol{x}_i) = \Phi(\boldsymbol{x}_i^\top \boldsymbol{\beta})$$

- **Random utility formulation**:

$$\begin{aligned} u_i &\sim N(\boldsymbol{x}_i^\top \boldsymbol{\beta}, 1) \\ y_i &= \begin{cases} 1 & \text{if } u_i > 0 \\ 0 & \text{if } u_i \leq 0 \end{cases} \end{aligned}$$

- Gibbs sampling samples from $p(\boldsymbol{\beta}, u_1, \ldots, u_n | \mathbf{y}, \mathbf{X})$:
  - Sample $\boldsymbol{\beta} | \mathbf{u}, \mathbf{y}, \mathbf{X}$ using linear regression update
  - Sample each $u_i | \boldsymbol{\beta}, \mathbf{y}, \mathbf{X}$ using truncated normal

Random walk Metropolis algorithm

- **Initialize** $\boldsymbol{\theta}^{(0)}$ and iterate for $i = 1, 2, \ldots$

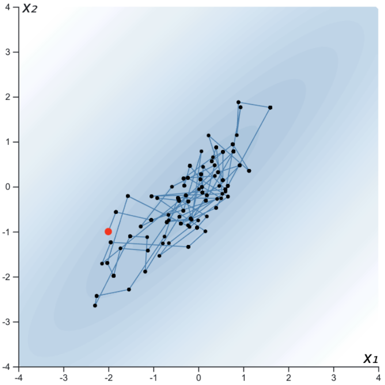
  1. **Sample proposal**: $\boldsymbol{\theta}_p | \boldsymbol{\theta}^{(i-1)} \sim N\left(\boldsymbol{\theta}^{(i-1)}, c \cdot \Sigma\right)$

  2. Compute the **acceptance probability**

  $$\alpha = \min\left(1, \frac{p(\mathbf{y}|\boldsymbol{\theta}_p)p(\boldsymbol{\theta}_p)}{p(\boldsymbol{\theta}^{(i-1)}|\mathbf{y})}\right)$$

  3. With probability α set $\boldsymbol{\theta}^{(i)} = \boldsymbol{\theta}_p$ and $\boldsymbol{\theta}^{(i)} = \boldsymbol{\theta}^{(i-1)}$ otherwise.

Random walk Metropolis



Number of iterations

250

Standard deviation of proposals

1

This parameter controls how far the sampler can jump in each iteration.

Random walk Metropolis, cont.

- Common choices of Σ in proposal $N\left(\theta^{(i-1)}, c \cdot \Sigma\right)$:
  - $\Sigma = I$ (proposes 'off the cigar')
  - $\Sigma = J_{\mathbf{y}}^{-1}(\hat{\theta})$ (propose 'along the cigar')
  - **Adaptive**. Start with $\Sigma = I$. Update Σ from initial run.

- Set c so average acceptance probability is 25-30%.

- **Good proposal**:
  - **Easy to sample**
  - **Easy to compute** α
  - Proposals should take reasonably **large steps** in θ-space
  - Proposals should **not be reject too often**.

The Metropolis-Hastings algorithm

- Generalization when the proposal density is not symmetric.

- Initialize $\theta^{(0)}$ and iterate for $i = 1, 2, \ldots$

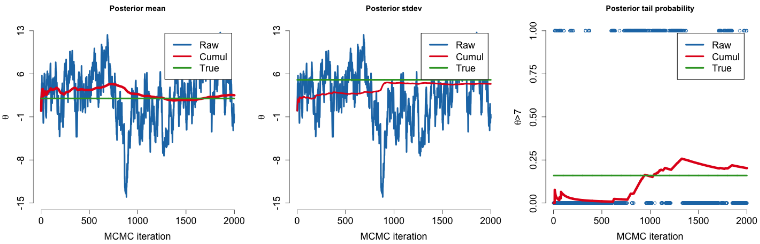
  1. **Sample proposal**: $\theta_p \sim q\left(\cdot|\theta^{(i-1)}\right)$
  2. Compute the **acceptance probability**

  $$\alpha = \min\left(1, \frac{p(\mathbf{y}|\theta_p)p(\theta_p)}{p(\mathbf{y}|\theta^{(i-1)})p(\theta^{(i-1)})} \frac{q\left(\theta^{(i-1)}|\theta_p\right)}{q\left(\theta_p|\theta^{(i-1)}\right)}\right)$$

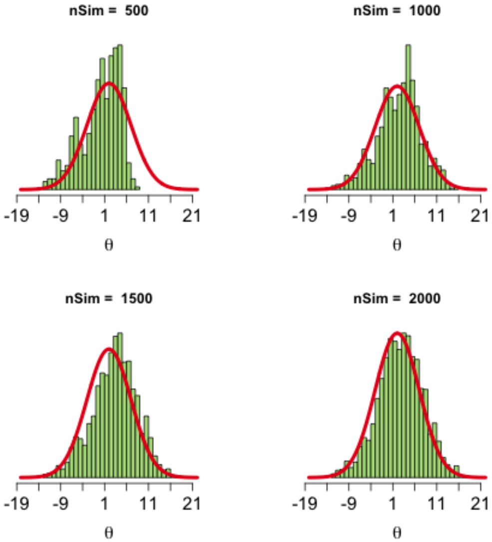
  3. With probability α set $\theta^{(i)} = \theta_p$ and $\theta^{(i)} = \theta^{(i-1)}$ otherwise.

Burn-in and convergence

- How long **burn-in**?
- **How long to sample** after burn-in?
- **Thinning**? Keeping every h draw reduces autocorrelation.
- **Convergence diagnostics**
  - Raw plots of simulated sequences (trajectories)
  - CUSUM plots
  - Check multiple runs with different initial values.
  - Potential scale reduction factor, `Rhat`.



Burn-in and convergence



Hamiltonian Monte Carlo

- When $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_p)^\top$ is **high-dimensional**, $p(\boldsymbol{\theta}|\mathbf{y})$ usually located in some subregion of $\mathbb{R}^p$ with complicated geometry.

- MH: hard to find good proposal distribution $q\left(\cdot|\boldsymbol{\theta}^{(i-1)}\right)$.

- MH: use very small step sizes otherwise too many rejections.

- **Hamiltonian Monte Carlo** (**HMC**):
  - distant proposals **and**
  - high acceptance probabilities.

- **HMC** is a **Metropolis-Hastings** + **proposal tailored to the posterior distribution** using ideas from Physics.

Hamiltonian Monte Carlo

- Physics: **Hamiltonian** system $H(\boldsymbol{\theta}, \boldsymbol{\phi}) = U(\boldsymbol{\theta}) + K(\boldsymbol{\phi})$, where U is the **potential energy** and K is the **kinetic energy**.
- **Hamiltonian Dynamics**

$$\frac{d\theta_i}{dt} = \frac{\partial H}{\partial \phi_i} = \frac{\partial K}{\partial \phi_i},$$
$$\frac{d\phi_i}{dt} = -\frac{\partial H}{\partial \theta_i} = -\frac{\partial U}{\partial \theta_i}$$

- Hockey puck sliding over a friction-less surface: illustration.
- **Posterior sampling**: $U(\boldsymbol{\theta}) = -\log\left[p(\boldsymbol{\theta})\, p(\mathbf{y}|\boldsymbol{\theta})\right]$.
- Momentum: $\boldsymbol{\phi} \sim N(\mathbf{0}, \mathbf{M})$ where $\mathbf{M}$ is the mass matrix and

$$K(\boldsymbol{\phi}) = -\log\left[p(\boldsymbol{\phi})\right] = \frac{1}{2}\boldsymbol{\phi}^\top \mathbf{M}^{-1}\boldsymbol{\phi} + \text{const}$$

- If we could propose $\boldsymbol{\theta}$ in continuous time (spoiler: we can't), the acceptance probability would be one.

Hamiltonian Monte Carlo

- **Hamiltonian Dynamics**

$$\frac{d\theta_i}{dt} = \left[\mathbf{M}^{-1}\boldsymbol{\phi}\right]_i,$$
$$\frac{d\phi_i}{dt} = \frac{\partial \log p\left(\boldsymbol{\theta}|\mathbf{y}\right)}{\partial \theta_i}$$

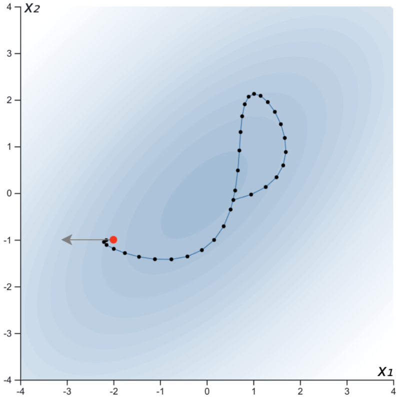
approximated using L steps with the **leapfrog algorithm**

$$\phi_i\left(t+\frac{\varepsilon}{2}\right) = \phi_i\left(t\right) + \frac{\varepsilon}{2}\frac{\partial \log p\left(\boldsymbol{\theta}|\mathbf{y}\right)}{\partial \theta_i}|_{\theta(t)}$$
$$\theta_i\left(t+\varepsilon\right) = \theta_i\left(t\right) + \varepsilon\mathbf{M}^{-1}\phi_i\left(t+\frac{\varepsilon}{2}\right),$$
$$\phi_i\left(t+\varepsilon\right) = \phi_i\left(t+\frac{\varepsilon}{2}\right) + \frac{\varepsilon}{2}\frac{\partial \log p\left(\boldsymbol{\theta}|\mathbf{y}\right)}{\partial \theta_i}|_{\theta(t+\varepsilon)},$$

where ε is the **step size**.

- **Discretization** $\Rightarrow$ acceptance probability drops with ε.

Leapfrog algorithm



Step size

0.2

How far the integrator moves at each step

Trajectory length

33

Number of steps to take in the trajectory

The Hamiltonian Monte Carlo algorithm

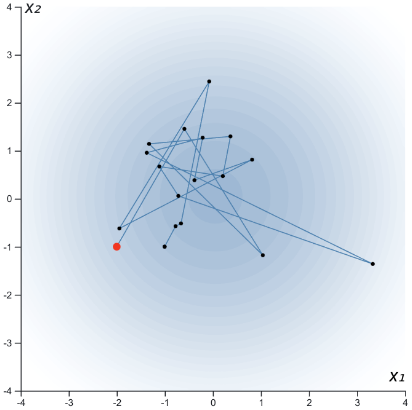
- Initialize $\theta^{(0)}$ and iterate for $i = 1, 2, \ldots$

  1. Sample the starting **momentum** $\phi_s \sim N(0, \mathbf{M})$
  2. Simulate new values for (θ_p, ϕ_p) by iterating the **leapfrog algorithm** L times with step size ε, starting in $\left(\theta^{(i-1)}, \phi_s\right)$.
  3. Compute the **acceptance probability**

  $$\alpha = \min\left(1, \frac{p(\mathbf{y}|\theta_p)p(\theta_p)}{p(\mathbf{y}|\theta^{(i-1)})p(\theta^{(i-1)})} \frac{p(\phi_p)}{p(\phi_s)}\right)$$

  4. With probability α set $\theta^{(i)} = \theta_p$ and $\theta^{(i)} = \theta^{(i-1)}$ otherwise.

Hamiltonian Monte Carlo



The Hamiltonian Monte Carlo algorithm

- Initialize $\theta^{(0)}$ and iterate for $i = 1, 2, \ldots$

  1. Sample the starting **momentum** $\phi_s \sim N(0, \mathbf{M})$
  2. Simulate new values for (θ_p, ϕ_p) by iterating the **leapfrog algorithm** L times with step size ε, starting in $\left(\theta^{(i-1)}, \phi_s\right)$.
  3. Compute the **acceptance probability**

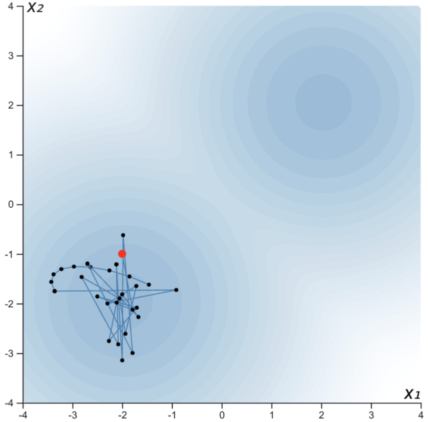
  $$\alpha = \min\left(1, \frac{p(\mathbf{y}|\theta_p)p(\theta_p)}{p(\mathbf{y}|\theta^{(i-1)})p(\theta^{(i-1)})} \frac{p\left(\phi_p\right)}{p\left(\phi_s\right)}\right)$$

  4. With probability α set $\theta^{(i)} = \theta_p$ and $\theta^{(i)} = \theta^{(i-1)}$ otherwise.

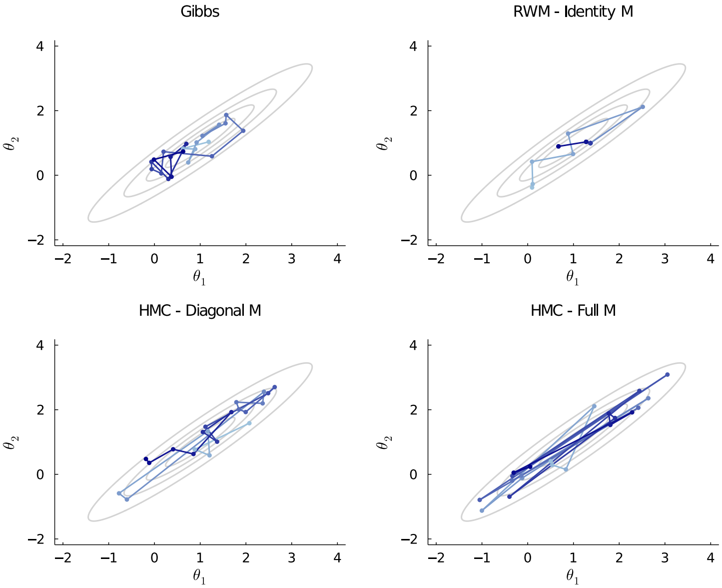
Tuning Hamiltonian Monte Carlo

- HMC is very efficient, but **needs careful tuning** to work.
- **Tuning parameters**:
  - **stepsize** ε,
  - **number of leapfrog** iterations L and
  - **mass matrix** M. (hello $J_{\mathbf{x}}^{-1}(\hat{\theta})$, my old friend)
- **No U-turn** sampler:
  - **Warm-up** to determine ε and L to get good acceptance rate.
  - Avoids U-turns in the Hamiltonian proposals.
- Drawbacks of HMC:
  - Need to **evaluate gradient of log posterior** many times during Hamiltonian iterations. Costly! (Subsampling HMC).
  - Difficulty with **multimodality** (true for most algorithms).
  - Standard HMC cannot handle **discrete parameters**. Mixture example. Some recent progress.

👀 Hamiltonian Monte Carlo on multimodal posterior



Comparing algorithms for bivariate normal



Comparing algorithms for bivariate normal