An Introduction to the Basic Math and Physics Behind Hamiltonian Monte Carlo

ChatGPT verified by Hedibert Lopes

May 15th 2025

1. Goal: Sampling from a Target Distribution

In Bayesian inference, we often need to sample from a posterior distribution $p(\theta)$, which may have a complex shape and be high-dimensional. Traditional MCMC methods (e.g., Metropolis-Hastings) can be inefficient due to their random walk behavior.

Hamiltonian Monte Carlo (HMC) is a more efficient sampling method that borrows ideas from classical physics to propose distant moves in parameter space with higher acceptance rates.

2. Hamiltonian Mechanics Basics

In classical mechanics, a system's state is described by:

- **Position:** θ (parameters to sample)
- **Momentum:** r (auxiliary variable)

The total energy of the system is given by the **Hamiltonian**:

$$H(\theta, r) = U(\theta) + K(r)$$

- $U(\theta)$ is the **potential energy**, defined as:

$$U(\theta) = -\log p(\theta)$$

- $K(r)$ is the **kinetic energy**, typically:

$$K(r) = \frac{1}{2} r^T M^{-1} r$$

where $r \sim \mathcal{N}(0, M)$ and M is a mass matrix (often the identity).

3. Hamilton's Equations

The evolution of the system is governed by Hamilton's equations:

$$\frac{d\theta}{dt} = \frac{\partial H}{\partial r} = M^{-1} r$$
$$\frac{dr}{dt} = -\frac{\partial H}{\partial \theta} = -\nabla_\theta U(\theta)$$

These equations describe how position and momentum evolve over time in a frictionless system.

4. Leapfrog Integration

Since Hamilton's equations cannot be solved analytically in most cases, we simulate the dynamics numerically using the **leapfrog integrator**, which preserves volume and is reversible:

1. Half-step update of momentum:
$$r \leftarrow r - \frac{\epsilon}{2} \nabla_\theta U(\theta)$$
2. Full-step update of position:
$$\theta \leftarrow \theta + \epsilon M^{-1} r$$
3. Another half-step update of momentum:
$$r \leftarrow r - \frac{\epsilon}{2} \nabla_\theta U(\theta)$$

Here, ϵ is the step size.

5. Metropolis Correction

After simulating the dynamics:

- Compute the Hamiltonian before and after the simulation:
$$H_{\text{old}} = U(\theta) + K(r), \quad H_{\text{new}} = U(\theta') + K(r')$$
- Accept the proposed state θ' with probability:
$$\min\left(1, \exp(H_{\text{old}} - H_{\text{new}})\right)$$

This step corrects for numerical error and ensures samples are from the correct distribution.

6. Summary Table

Concept	**Meaning**
θ	Parameter (position)
r	Momentum
$U(\theta)$	Potential energy (log-probability)
$K(r)$	Kinetic energy
$H(\theta, r)$	Total energy
Leapfrog integrator	Numerical method to simulate dynamics
Metropolis step	Accept/reject mechanism

7. Why HMC Works

HMC makes use of gradient information to propose efficient transitions across the parameter space. This reduces the random walk behavior common in traditional MCMC and makes HMC especially effective in high-dimensional problems.

7 Simple HMC Demo in R - Drawing θs from $N(0, 1)$

```
# Potential energy (negative log-probability)
U <- function(theta) {
  return(0.5 * theta^2)
}

# Gradient of potential energy
grad_U <- function(theta) {
  return(theta)
}

# Kinetic energy
K <- function(r) {
  return(0.5 * r^2)
}

# Leapfrog integrator
leapfrog <- function(theta, r, epsilon, L) {
  r <- r - 0.5 * epsilon * grad_U(theta)
  for (i in 1:L) {
    theta <- theta + epsilon * r
    if (i != L) {
      r <- r - epsilon * grad_U(theta)
    }
  }
  r <- r - 0.5 * epsilon * grad_U(theta)
  return(list(theta = theta, r = r))
}
```

```
# HMC sampler
hmc <- function(theta0, epsilon, L, n_samples) {
  samples <- numeric(n_samples)
  theta <- theta0

  for (i in 1:n_samples) {
    r0 <- rnorm(1)  # sample momentum
    current_H <- U(theta) + K(r0)

    # Simulate Hamiltonian dynamics
    lf <- leapfrog(theta, r0, epsilon, L)
    theta_prop <- lf$theta
    r_prop <- lf$r
    proposed_H <- U(theta_prop) + K(r_prop)

    # Metropolis acceptance
    accept_prob <- exp(current_H - proposed_H)
    if (runif(1) < accept_prob) {
      theta <- theta_prop  # accept
    }
    samples[i] <- theta
  }

  return(samples)
}

# Run the HMC sampler
set.seed(42)
samples <- hmc(theta0 = 0, epsilon = 0.1, L = 10, n_samples = 5000)

# Plot the results
hist(samples, probability = TRUE, breaks = 40, col = "skyblue",
     main = "HMC Samples from N(0, 1)", xlab = "theta")
curve(dnorm(x), col = "red", lwd = 2, add = TRUE)
legend("topright", legend = c("True density", "HMC samples"),
       col = c("red", "skyblue"), lty = 1, lwd = 2, bty = "n")
```