STAT6440 Final Project: Hamiltonian Monte Carlo with Reflection and Refraction

Name: Ruizhong Miao
ID: rm9dd

1 Introduction

Markov Chain Monte Carlo (MCMC) is a widely used algorithm in Bayesian statistics. When the target distribution is continuous, Hamiltonian Monte Carlo (HMC) adds an auxiliary variable, namely momentum, and simulates Hamiltonian dynamics. Compared with MCMC, HMC often leads to a more rapid exploration of the distribution. The most commonly used algorithm in HMC is the leapfrog algorithm. The leapfrog algorithm makes use of the gradient of the target distribution. When the target distribution is only piecewise differentiable, the leapfrog algorithm is no longer applicable. To extend the usage of HMC to piecewise continuous distribution, Afshar[1] introduced a modification of the Leapfrog discretization, called Reflective Hamiltonian Monte Carlo (RHMC), in which the momentum is reflected or refracted at the intersection of the trajectory and discontinuity. In their algorithm, when the trajectory meets the boundaries between two piecewise continuous regions, the momentum is decomposed into two components, one perpendicular to the boundary and one parallel to the boundary. Then, the perpendicular component is either reflected or refracted depending on whether the momentum is large enough to climb over the energy gap. Figure 1 illustrates the process.

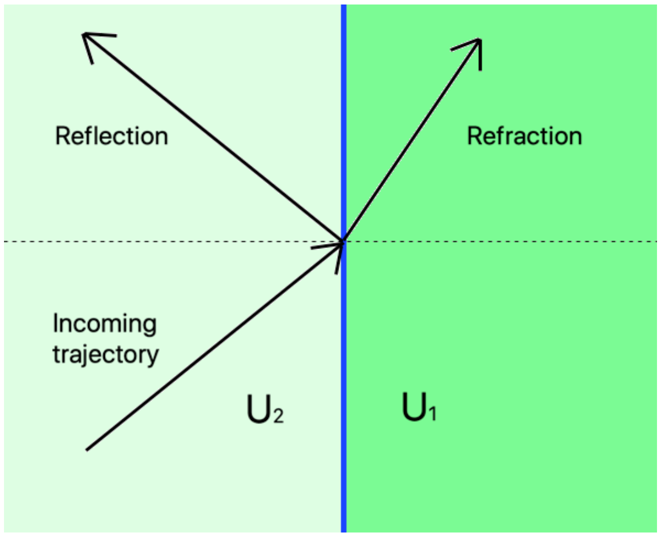


Figure 1: Reflection and refraction

In this project, we make another modification to the algorithm proposed in Afshar[1]: We make use of the fact that Hamiltonian dynamics is followed along each direction separately.

Therefore, we don't have to decompose the momentum into the perpendicular and parallel components. Instead, we can simulate Hamiltonian dynamics separately within each dimension.

2 Method

2.1 Hamiltonian dynamics and leapfrog algorithm

In Hamiltonian dynamics, a physical system is described by a set of canonical coordinates $(\mathbf{q}, \mathbf{p})$, where $\mathbf{q}$ and $\mathbf{p}$ are position and momentum respectively. Define the system's Hamiltonian

$$H(\mathbf{q}, \mathbf{p}) = U(\mathbf{q}) + K(\mathbf{p}) = U(\mathbf{q}) + \frac{\mathbf{p}^2}{2m},$$

where $U(\mathbf{q})$ is the potential energy and is a function of $\mathbf{q}$, and $K(\mathbf{p}) = \frac{\mathbf{p}^2}{2m}$ is the kinetic energy and is a function of $\mathbf{p}$. m is the mass of the physical object we are describing, and we set $m = 1$ in HMC. The time evolution of the system is described by the following two equations

$$\frac{d\mathbf{p}}{dt} = -\frac{\partial H(\mathbf{q}, \mathbf{p})}{\partial \mathbf{q}} = -\frac{\partial U(\mathbf{q})}{\partial \mathbf{q}},$$
$$\frac{d\mathbf{q}}{dt} = \frac{\partial H(\mathbf{q}, \mathbf{p})}{\partial \mathbf{p}} = \frac{\partial K(\mathbf{p})}{\partial \mathbf{p}}.$$

Suppose ϵ is a small constant. The leapfrog algorithm finds the trajectory of $(\mathbf{q}, \mathbf{p})$ by iteratively updating $\mathbf{q}$ and $\mathbf{p}$ as follows:

$$p_i(t + \epsilon/2) = p_i(t) - \epsilon/2\,\nabla_{q_i} U(\mathbf{q}(t)),$$
$$q_i(t + \epsilon) = q_i(t) + \epsilon\,\nabla_{p_i} K(\mathbf{p}(t + \epsilon/2)) = q_i(t) + \epsilon p_i(t + \epsilon/2),$$
$$p_i(t + \epsilon) = p_i(t + \epsilon/2) - (\epsilon/2)\,\nabla_{q_i} U(\mathbf{q}(t + \epsilon)).$$

In one iteration of HMC, the above procedure is repeated several times to get a trajectory of $(\mathbf{q}, \mathbf{p})$.

Another law all physical systems follow is the conservation of energy. Suppose $(\mathbf{q}_0, \mathbf{p}_0)$ is the initial state and $(\mathbf{q}_1, \mathbf{p}_1)$ is the end state of the trajectory. Then we have

$$U(\mathbf{q}_0) + K(\mathbf{p}_0) = U(\mathbf{q}_1) + K(\mathbf{p}_1).$$

However, due to the discretization error of the leapfrog algorithm, the the above equation does not always hold. We introduce the following acceptance probability to account for the discretization errors:

$$\exp(U(\mathbf{q}_0) + K(\mathbf{p}_0) - U(\mathbf{q}_1) - K(\mathbf{p}_1)).$$

Then, at the end of the HMC iteration, we either accept $(\mathbf{q}_1, \mathbf{p}_1)$ or keep $(\mathbf{q}_0, \mathbf{p}_0)$ as the new state. It is proven that the $\mathbf{q}$ component of the resulting Markov chain will have a distribution proportional to $\exp(-U(\mathbf{q}))$.

2.2 Reflection and Refraction

The leapfrog algorithm requires the gradients of the potential energy. If the potential is piecewise continuous, the gradient is not available at the boundary of two discontinuous regions. However, the conservation of energy still holds:

$$U(\mathbf{q}(t)) + K(\mathbf{p}(t)) = U(\mathbf{q}(t+\delta)) + K(\mathbf{p}(t+\delta)).$$

More importantly, in a physical system, the conservation of energy holds along each direction, which means

$$U(q_i(t)) + K(p_i(t)) = U(q_i(t+\delta)) + K(p_i(t+\delta)),$$

$$U(q_i(t)) + \frac{p_i(t)^2}{2} = U(q_i(t+\delta)) + \frac{p_i(t+\delta)^2}{2} \text{ for } i = 1,2,...,n.$$

We can use this to update $(\mathbf{q}, \mathbf{p})$. We can rewrite the above equation as

$$\frac{p_i(t)^2}{2} = U(q_i(t+\delta)) - U(q_i(t)) + \frac{p_i(t+\delta)^2}{2},$$

$$\frac{p_i(t)^2}{2} = \Delta U + \frac{p_i(t+\delta)^2}{2},$$

where $\Delta U = U(q_i(t+\delta)) - U(q_i(t))$ is the gap in potential energy between initial position and end position. In a physical system, if the initial kinetic energy of an object is less than ΔU, the object will bounce off the energy barrier, resulting in a reflection. In this case, we can update p_i by flipping its sign. On the other hand, if the initial kinetic energy of an object is greater than ΔU, we can update p_i by

$$p_i(t+\delta) = sign(p_i(t))\sqrt{2\Delta U - p_i(t)^2}.$$

We summarize our algorithm in the following:

```
Input: Initial position $\mathbf{q}(t)$, number of steps $L$, step size $\epsilon$, variance for initial momentum $\sigma^2$.
for $i = 1,2,...,n$:
    Initialize momentum: $p_i(t) \sim N(0,\sigma^2)$.
for $l = 0,1,...,L-1$:
    for $i = 1,2,...,n$:
        Make an attempt move: $q_i^{attempt} = q_i(t + l\delta) + \epsilon p_i(t + l\delta)$.
        Calculate the potential energy gap: $\Delta U = U(q_i^{attempt}) - U(q_i(t + l\delta))$.
        If $\frac{p_i(t + l\delta)^2}{2} \leq \Delta U$:
                Update: $p_i(t+(l+1)\delta) = -p_i(t+l\delta)$.
        else:
                Update: $q_i(t+(l+\frac{1}{2})\delta) = q_i(t) + \frac{\epsilon}{2} p_i(t+l\delta)$.
                Update: $p_i(t+(l+1)\delta) = sign(p_i(t+l\delta))\sqrt{2\Delta U - p_i(t+l\delta)^2}$.
                Update: $q_i(t+(l+1)\delta) = q_i(t+(l+\frac{1}{2})\delta) + \frac{\epsilon}{2} p_i(t+(l+1)\delta)$.
Output: $\mathbf{q}(t+L\delta)$, $\mathbf{p}(t+L\delta)$.
```

Note that the above procedure does not require the gradient of the potential energy. Therefore it is applicable for piecewise continuous distributions. We accept the final state with acceptance probability $\exp(U(\mathbf{q}(t)) + K(\mathbf{p}(t)) - U(\mathbf{q}(t+L\delta)) - K(\mathbf{p}(t+L\delta)))$.

3 Experiment:

3.1 Simulating samples from target distribution

To test the correctness of our algorithm, we first simulate samples from the following two distributions

- Standard normal distribution: The probability density function and the corresponding potential energy are

$$p(q) \propto e^{-\frac{q^2}{2}},\ U(q) = \frac{q^2}{2}.$$

- Step function density: The probability density function and the corresponding potential energy are

$$p(q) \propto \begin{cases} 1, & q \in [0,1] \\ 2, & q \in [1,2] \\ 0, & \text{otherwise} \end{cases},\ U(q) \propto \begin{cases} -\log(1), & q \in [0,1] \\ -\log(2), & q \in [1,2] \\ \infty, & \text{otherwise} \end{cases}.$$

The histograms of the samples generated from the two distributions are

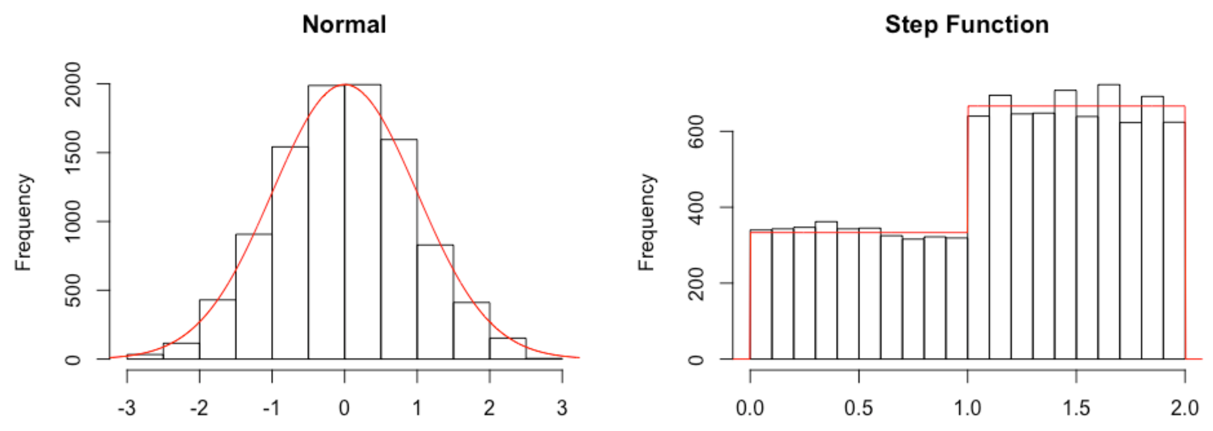


Figure 2: Simulated samples

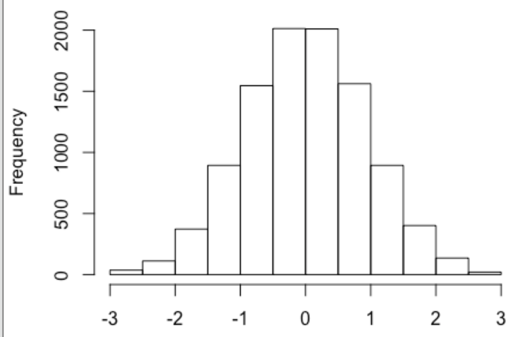
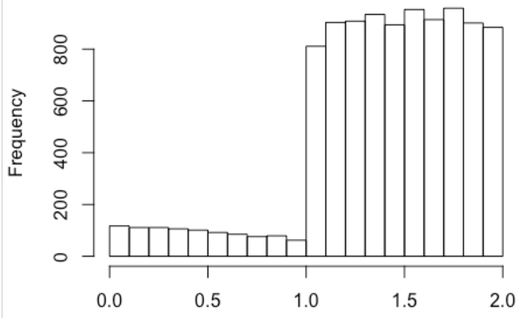
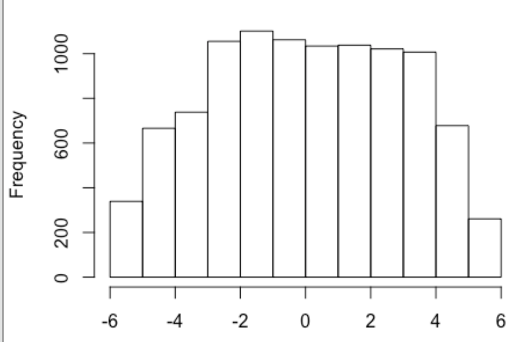
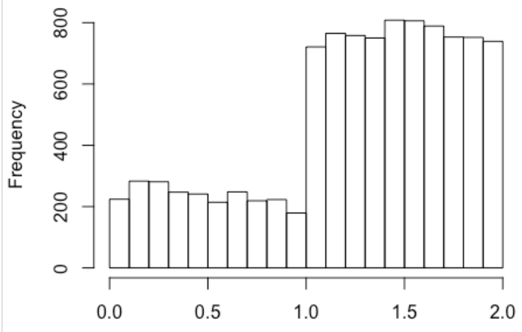
The red curves are the true theoretical distribution. We can see that our algorithm successfully generates samples that follow the target distributions.

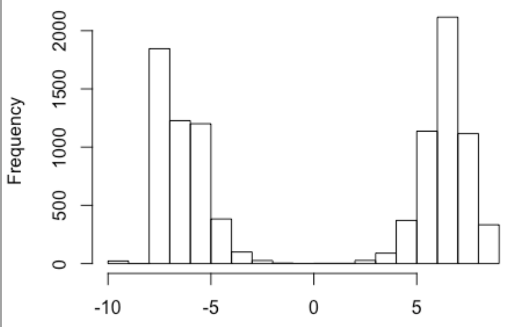
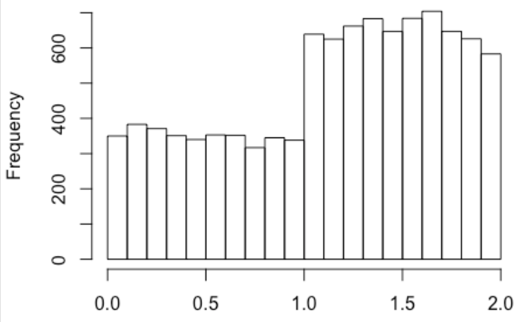
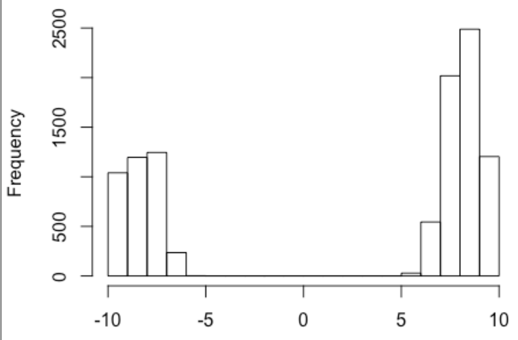
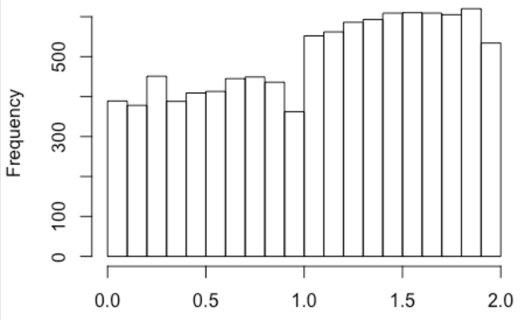
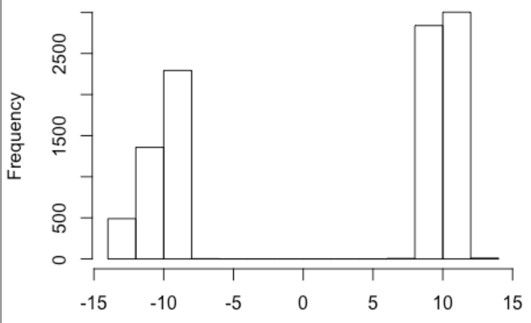
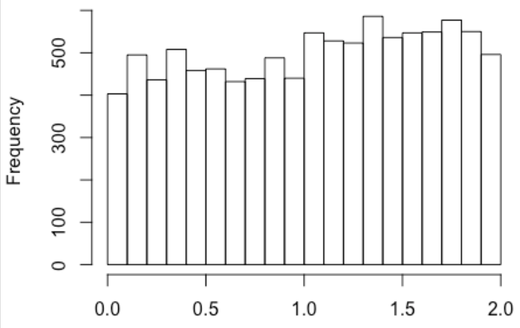
3.2 Choice of momentum

During each iteration of HMC, the momentum is generated from $p_i(t) \sim N(0,\sigma^2)$. The parameter σ^2 determines the "spread" of the normal distribution, and therefore the magnitude of the momentum. If σ^2 is too small, the momentum will be small, and the trajectory may not have enough kinetic energy to climb over the potential energy gap; If σ^2 is too large, the momentum may also get too large, and the step size of one HMC iteration will be large. This can lead to undesirable results, as we demonstrate below.

For the two distributions in the previous subsection, we demonstrate the impact of σ^2 by plotting the histograms of samples generated under different values of σ^2. The following table shows the results:

Table 1: The effect of initial momentum

	Normal Distribution	Step Function
σ^2=1		
σ^2=1.5		

	Normal Distribution	Step Function
σ^2=2		
σ^2=2.5		
σ^2=3		

We can see that for the normal distribution, when σ^2 is too large, the generated points will concentrate on the two tails. The reason is that if the momentum is too large, the trajectory will bounce back and forth between the two tails, and will not settle in the middle. For the stepwise function, when σ^2 is too small, most samples will concentrate on $[1,2]$, because the kinetic energy is not high enough to overcome the potential energy gap. When σ^2 is too large, the potential energy gap is negligible compared to the large kinetic energy, so the samples behave like uniform random variables.

3.3 Comparison to random walk Monte Carlo algorithm

The advantage of HMC over the random walk Monte Carlo (RWMC) is HMC can explore the entire distribution faster, especially when the target distribution is multimodal. To compare HMC to RWMC, we design the following distribution

$$p(q_1, q_2) \propto \begin{cases} \exp(q_1^2 + q_2^2), \text{ for } q_1 \in [-1,1] \text{ and } q_2 \in [-1,1] \\ 0, \text{ otherwise} \end{cases}.$$

The potential energy it corresponds to is

$$U(q_1, q_2) \propto \begin{cases} -q_1^2 - q_2^2, \text{ for } q_1 \in [-1,1] \text{ and } q_2 \in [-1,1] \\ \infty, \text{ otherwise} \end{cases}.$$

We simulate samples from this distribution using both HMC and RWMC. The following two figures show the samples:

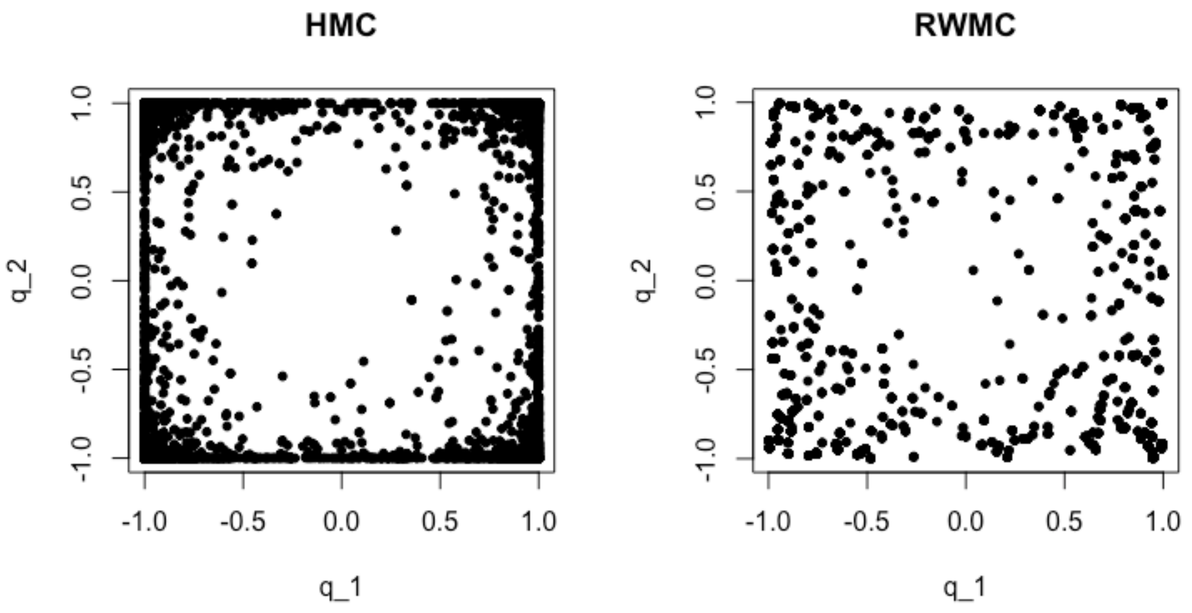


Figure 3: Comparing HMC to RWMC

In this simulation, the number of samples are the same for HMC and RWMC. From the figures we can see that there are more distinct points in the first figure. This is because RWMC has a higher chance of rejecting a proposed point, so a lot of points in the simulated samples in RWMC are repetitive values. In comparison, HMC accepts more proposed points, and explores the entire distribution more efficiently.

Conclusion

In this project, we implemented a HMC algorithm that incorporated reflection and refraction. This algorithm can sample from piecewise continuous distribution, in comparison to the leapfrog algorithm, which can only sample from continuous distribution. We also found that the initial momentum during each HMC iteration can affect the target distribution of the algorithm, and the parameters governing its magnitude need careful tunings. We also demonstrated the efficiency of HMC in exploring the entire target distribution compared to RWMC.

Reference

[1] Afshar, Hadi Mohasel, and Justin Domke. "Reflection, refraction, and hamiltonian monte carlo." Advances in Neural Information Processing Systems. 2015.