Inference and Representation, Fall 2017

Problem Set 5: Hamiltonian Monte-Carlo

Zhuoru Lin
zlin@nyu.edu

Disclaimer: I adhered to NYU honor code in this assignment.

1. To show $\mathcal{L}(\mathbf{v}, \mathbf{x})$ is convex given $\mathbf{v}$, we show the second derivative of $\mathcal{L}$ with respect to $\mathbf{v}$ is larger than zero for all $\mathbf{v}$. In fact:

$$\frac{\partial^2 \mathcal{L}}{\partial \mathbf{v}^2} = M. \tag{1}$$

By definition M is a positive matrix. Therefore $\mathcal{L}$ is convex with respect to $\mathbf{v}$.

2. We use the fact that the pointwise suprenum of a set of convex functions is also convex. Consider any $y \in \Omega$, we must have $< y, p > +f(y)$ is convex with respect to p, therefore the convex conjugate:
$$f^*(p) = \sup_y (< y, p > +f(x)) \tag{2}$$
is convex.

3. By the definition of $\mathcal{L}$ and $\mathbf{p}$ we have:

$$\mathbf{p} = \frac{\partial \mathcal{L}}{\partial \mathbf{v}} = M\mathbf{v}. \tag{3}$$

The convex conjugate of $\mathcal{L}(\mathbf{v})$ is:

$$\begin{aligned}
\mathcal{L}^*(v) &= \sup_v [< v, p > - \mathcal{L}(\mathbf{v})] \\
&= \sup_v [< v, Mv > - \frac{1}{2} < v, mv > + U(\mathbf{x})] \\
&= \sup_v [\frac{1}{2} < v, mv > + U(\mathbf{x})]
\end{aligned}$$

There fore the Hamilton $\mathcal{H}$ must have the form:

$$\mathcal{H}(\mathbf{p}, \mathbf{x}) = \frac{1}{2} < \mathbf{p}, M^{-1}\mathbf{p} > + U(\mathbf{x}) \tag{4}$$

4. By the definition of $\mathcal{H}$ (equation 4), we have:

$$\frac{\partial \mathcal{H}}{\partial \mathbf{p}} = M^{-1}\mathbf{p} = \mathbf{v}. \tag{5}$$

By the definition of $\mathcal{H}$ and $\mathcal{L}$ we also have:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{x}} = -\frac{\partial U}{\partial \mathbf{x}} = -\frac{\partial \mathcal{H}}{\partial \mathbf{x}} \tag{6}$$

By Euler-Lagrange equation and the definition of $\mathbf{p}$

$$\frac{\partial \mathcal{H}}{\partial \mathbf{x}} = -\frac{\partial \mathcal{L}}{\partial \mathbf{x}} - = \frac{d}{dt}\frac{\partial \mathcal{L}}{\partial \mathbf{v}} = -\frac{d\mathbf{p}}{dt}. \tag{7}$$

Equation 5 and 7 are *Hamiltonian Equations.*

5. Without loss of generality, let the temperature $T = 1$. The canonical distribution of hamiltonian $\mathcal{H}(\mathbf{x}, \mathbf{p})$ is:

$$\begin{aligned} p(\mathbf{x}, \mathbf{p}) &= \frac{1}{Z} \exp[-\mathcal{H}(\mathbf{x}, \mathbf{p})] \qquad (8) \\ &= \frac{1}{Z} \exp[-\frac{1}{2}\mathbf{p}^T M^{-1} \mathbf{p} - U(\mathbf{x})] \\ &= \frac{1}{Z} \exp[-\frac{1}{2}\mathbf{p}^T M^{-1} \mathbf{p}] \exp[-U(\mathbf{x})], \qquad (9) \end{aligned}$$

where Z is the partition function. Observed that $p(\mathbf{x}, \mathbf{p})$ is seperable function, therefore one can factor $p(\mathbf{x}, \mathbf{p}) = p(\mathbf{x})p(\mathbf{p})$.

6. From equation 9, we can derive that the expression of marginal distribution:

$$p(\mathbf{p}) = \int p(\mathbf{x}, \mathbf{p}) d\mathbf{x} \propto \exp[-\mathbf{p}^T M^{-1} \mathbf{p}]. \tag{10}$$

   From equation 10, we know that $p(\mathbf{p})$ is a Gaussian distribution with mean $\mathbf{0}$ and Varaince M.

7. The update of $\mathbf{x}$ and $\mathbf{p}$ is based on *Hamiltonian equation*, which is depended only on the hamiltonian $\mathcal{H}(\mathbf{x}, \mathbf{p})$ but not the partition function Z. Therefore the leapfrog algorithm does not require the knowledge of the partition function.

8. As shown in Radford's introduction of Hamiltonian MCMC [1], the HMC step leaves canonical distribution invariante. Also we know from previous problems that the canonical distribution is seperable. Therefore at any state of $\mathbf{x}$ and $\mathbf{p}$ during HMC steps the marginal distribution $p(\mathbf{x}) = \int p(\mathbf{x}, \mathbf{p}) d\mathbf{p} = p(\mathbf{x}) \int p(\mathbf{p}) d\mathbf{p} = p(\mathbf{x})$.

9. Before HMC, we first need to define the function to calculate potential energy $U(\mathbf{x})$ and the gradient at x position $\nabla_{\mathbf{x}} U(\mathbf{x})$. These correspond to **U** and **grad_U** in codes:

```
import numpy as np
from scipy.stats import multivariate_normal
from numpy.linalg import inv
from numpy.random import normal
from functools import partial
# Implement HMC for Q9 of HW5.
# Global variable
MEAN = np.array([0.0, 0.0])
COV_MAT = np.array([[1, 0.998], [0.998, 1]])

def U(x):
    x = np.array(x)
    p_x = multivariate_normal.pdf(x, mean=MEAN, cov=COV_MAT)
    return -np.log(p_x)

def grad_U(x):
    """
    Return partial derivative dU/dx given x
    """
    x = np.array(x)
    return inv(COV_MAT).dot(x)
```

Noticed that I used the fact that $\nabla_{\mathbf{x}} U(\mathbf{x}) = \Sigma^{-1} x$ with $U(x) = -\log p(\mathbf{x})$ and $p(\mathbf{x})$ follows multivariate normaldistribution with zero mean and Σ covariance.

Then the implementation of HMC step is:

```
def HMC2(findE, gradE, epsilon, L, x):
    g = gradE ( x )  # set gradient using initial x
    E = findE ( x )
    p = normal(0, 1, len(x))
    H = p.dot(p) / 2 + E ; # evaluate H(x,p)
    xnew = x
    gnew = g
    for l in range(L): # make Tau 'leapfrog' steps
        p = p - epsilon * gnew / 2 ; # make half-step in p
        xnew = xnew + epsilon * p ; # make step in x
        gnew = gradE ( xnew ) ; # find new gradient
        p = p - epsilon * gnew / 2 ; # make half-step in p
    Enew = findE ( xnew ) ; # find new value of H
    Hnew = p.dot(p) / 2 + Enew ;
    dH = Hnew - H ; # Decide whether to accept
    if ( dH < 0 ):
        accept = 1
    elif ( np.random.uniform(0,1) < np.exp(-dH) ):
        accept = 1
    else:
```

```
        accept = 0 ;
    if accept:
        g = gnew
        x = xnew
        E = Enew
    return x
```

The result trajecgory is shown in Figure 1:

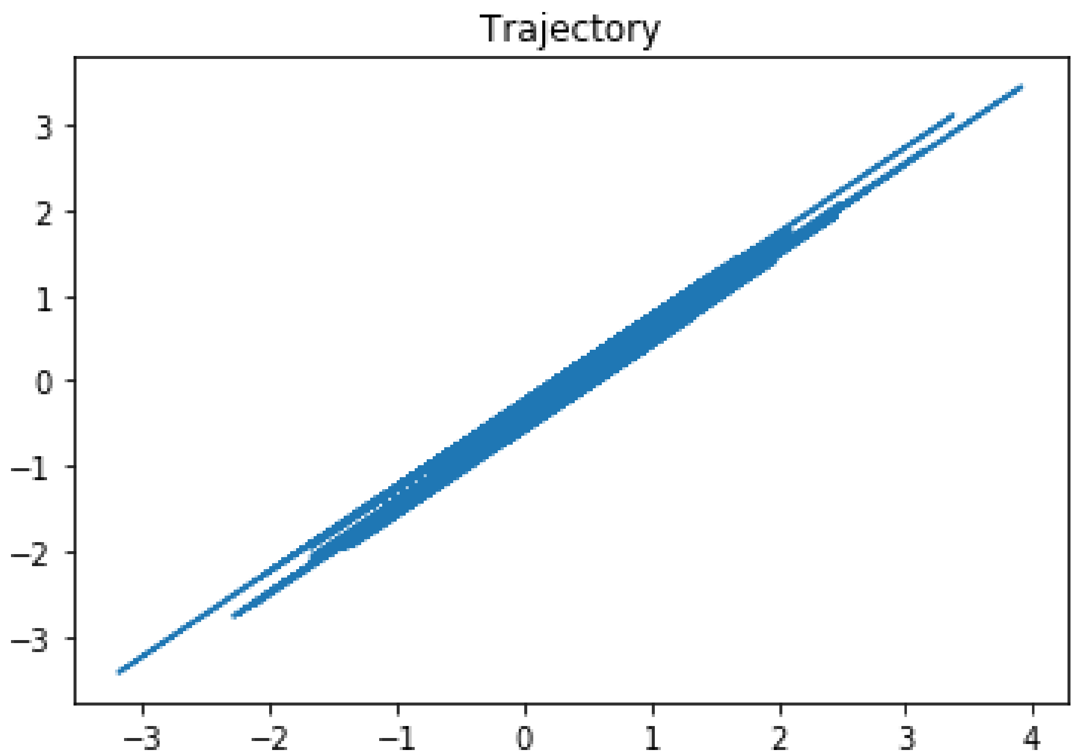

Figure 1: The trajectory of Hamiltonian HMC sampled from bivariate normal distribution with correlation 0.998.

References

[1] Radford M Neal et al. Mcmc using hamiltonian dynamics. *Handbook of Markov Chain Monte Carlo*, 2(11), 2011.