Q2 Langevin

February 25, 2024

0.1 Question 2: Sampling from energy models with Langevin dynamics and stein scores

Energy based models learn an energy functional $E_\theta : \mathcal{X} \to \mathbb{R}$. We look at the Gibbs distribution as follows:

$p_\theta(x) = \frac{1}{Z_\theta} e^{-E_\theta(x)}$, where $Z_\theta = \int_{\mathcal{X}} e^{-E_\theta(y)} dy$.

Directly sampling from p_θ is hard, but we can approximate samples using a Markov chain with stationary distribution p_θ, spscifically, we have the discretized Langevin dynamics:

$x_{t+1} = x_t - \eta \nabla_x \log p_\theta(x_t) + \sqrt{2\eta}\epsilon_t$,

where $\epsilon_t \sim \mathcal{N}(0, I)$, η is the step size.

We consider a 2D case, where $x \in \mathbb{R}^2$. Say $E_\theta(x) = \theta \cdot x$, where $\theta \in \mathbb{R}^2$ is a vector and has all the parameters.

Calculate the expression for the distribution x_N, where $x_0 \sim \mathcal{N}(0, I)$, and N is the number of steps, in terms of η, θ, N.

(You can implement and see if your computational results match your analytical results. A helpful website: https://courses.cs.washington.edu/courses/cse599i/20au/resources/L16_ebm.pdf)

Answer: We have

$$-\eta \nabla_x \log(p_\theta(x_t)) = -\eta \nabla_x \log(\frac{1}{Z_\theta} e^{-E_\theta(x_t)}) = -\eta \nabla_x (-E_\theta(x_t) - \log(Z_\theta)) = -\eta \nabla_x (-\theta \cdot x_t) = \eta\theta$$

So we have

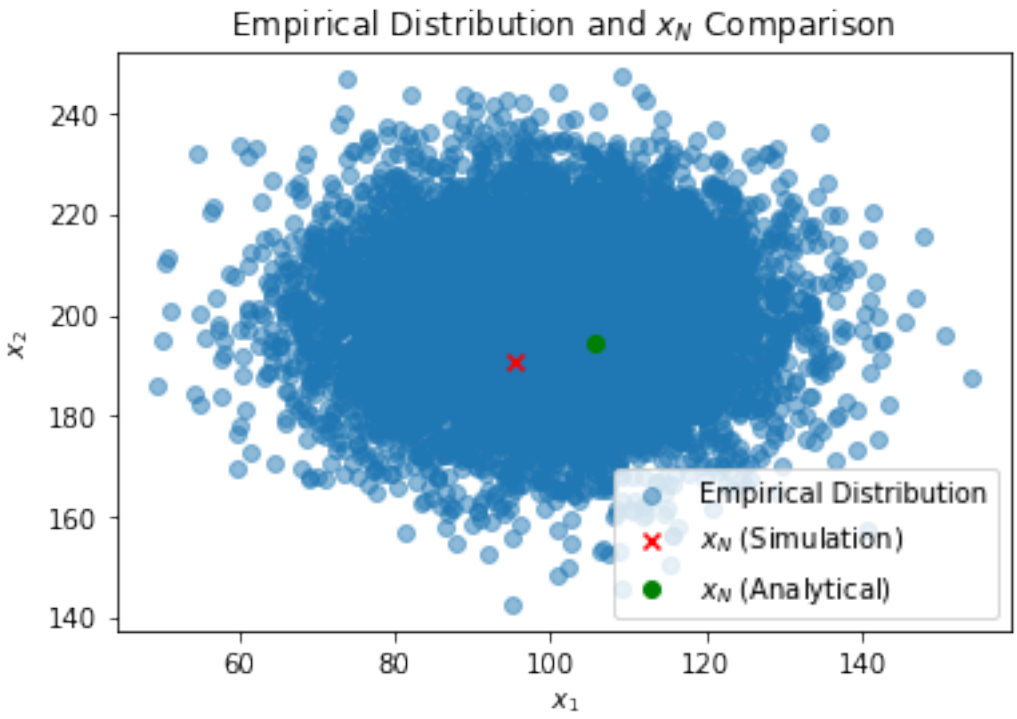
$$x_1 = x_0 + \eta\theta + \sqrt{2\eta}\varepsilon_0$$
$$x_2 = x_1 + \eta\theta + \sqrt{2\eta}\varepsilon_1 = x_0 + \eta\theta + \sqrt{2\eta}\varepsilon_0 + \eta\theta + \sqrt{2\eta}\varepsilon_1$$
$$\vdots$$
$$x_N = x_0 + N\eta\theta + \sqrt{2\eta}\sum_{i=0}^{N} \varepsilon_i$$

Where $\sum_{i=0}^{N} \varepsilon_i \sim \mathcal{N}(0, NI)$

[ ]:
```
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import multivariate_normal
```

I'm sufficiently confident that this is right

[ ]:
```
def langevin_dynamics(x, theta, eta, N):
    for _ in range(N):
        gradient = -theta
        x = x - eta * gradient + np.sqrt(2 * eta) * np.random.normal(size=x.
 ↪shape)
    return x

# Parameters
theta = np.array([1.0, 2.0])  # Parameters of the energy function
eta = 0.1  # Step size
N = 1000  # Number of steps
num_samples = 10000  # Number of samples to generate for empirical distribution

# Initial condition
x_0 = np.random.normal(size=(2,))

# Simulate Langevin dynamics
x_N_simulation = langevin_dynamics(x_0, theta, eta, N)

# Analytical expression for x_N
x_N_analytical = x_0 + N * eta * theta + np.sqrt(2 * eta * N) * np.random.
 ↪normal(size=x_0.shape)

# Generate samples for empirical distribution
empirical_samples = np.array([langevin_dynamics(x_0, theta, eta, N) for _ in␣
 ↪range(num_samples)])

# Plot the results
plt.scatter(empirical_samples[:, 0], empirical_samples[:, 1], label='Empirical␣
 ↪Distribution', alpha=0.5)
plt.scatter(x_N_simulation[0], x_N_simulation[1], color='red', marker='x',␣
 ↪label='$x_N$ (Simulation)')
plt.scatter(x_N_analytical[0], x_N_analytical[1], color='green', marker='o',␣
 ↪label='$x_N$ (Analytical)')
plt.title('Empirical Distribution and $x_N$ Comparison')
plt.xlabel('$x_1$')
plt.ylabel('$x_2$')
plt.legend()
plt.show()
```


Empirical Distribution and x_N Comparison
240
220
200
180
160
140
x_2
60
80
100
120
140
x_1
Empirical Distribution
x_N (Simulation)
x_N (Analytical)