Likelihood Estimation Comparison for Energy Based Models: A Framework

Manuel Günther

March 18, 2024

Contents

1 Introduction

Energy based models are widely used in statistical learning and can also be found as elegant models of spatial interactions in point process models. However, maximum likelihood (ML) based inference usually remains challenging due to the difficulty in approximating the partition function using Markov Chain Monte Carlo (MCMC) for high- dimensional multimodal distributions.

Following the work of (), the goal of this project is to explore the efficacy and efficiency of recovery likelihood estimation in learning parametrised models. The RL approach is based on applying Gaussian noise to the training data and subsequently using the conditional instead of the marginal distribution. We will shown that this approach yields an unbiased estimator of the underlying parameters and investigate the MCMC sampling properties.

As the inference process depends on many components and hyperparameters, comparing these estimators empirically is necessarily not comprehensive. Depending on the training data, the model used, the sampling approach and components like the optimiser, the results might be different. Thus a large part of this project was dedicated to develop a general code framework that allows for more systematic testing in a setting of interest.

Gao and Song have shown that RL produces good results and computational savings in the context of image generation. In that context and many others the models are typically some variety of neural networks. For neural network architectures the concrete values of the weights and biases are less significant, and two architectures with fundamentally different parameter values can still produce similar distributions. In this project the emphasis is on parametric models whose parameters carry a direct meaning or function as the basis for further analysis. For such models it is not just important that the resulting distribution coincides with the data distribution, but also that the parameters themselves are estimated accurately. The later comparisons will hence focus primarily on how well the estimated parameters match the distribution parameters.

1.1 Problem Formulation

The problem that is investigated in statistical learning of generative models looks like this: We have a dataset $\mathcal{D} = (\boldsymbol{x}_i)_{i=1}^n \subseteq \mathbb{R}^d$ from an unknown distribution of a random vector $\boldsymbol{X} : \Omega \to \mathbb{R}^d$, with some probability density function $p : \mathbb{R}^d \to \mathbb{R}$. We want to use the data to approximate this density function with a parametrised model density function $p_{\boldsymbol{\theta}} : \mathbb{R}^d \to \mathbb{R}$, for a vector of parameters $\boldsymbol{\theta} \in \mathbb{R}^m$, by estimating an optimal $\hat{\boldsymbol{\theta}}$ such that $p_{\hat{\boldsymbol{\theta}}}$ is close to p.

1.2 Approaches

For high dimensional multimodal problems and complicated target densities p estimating $\hat{\boldsymbol{\theta}}$ is a highly non-trivial task, where the standard ML inference can be computationally infeasible. Complicating the problem further is that the model distributions need to be sufficiently complex to be able to resemble the target distribution and thus even the model density is often only determined by its probability kernel, as the normalisation constant is not available analytically and costly to calculate numerically.

There are various approaches that attempt to make this inference problem tractable, such as

2 Energy Based Models

For a function $U_{\boldsymbol{\theta}} : \mathbb{R}^d \to \mathbb{R}$, also called (potential) energy functional, an energy-based model (EBM) is defined as:

$$p_{\boldsymbol{\theta}}(\boldsymbol{x}) = \frac{1}{Z_{\boldsymbol{\theta}}} \exp(-U_{\boldsymbol{\theta}}(\boldsymbol{x}))$$

where $Z_{\boldsymbol{\theta}} = \int \exp(-U_{\boldsymbol{\theta}}(\boldsymbol{x}))\,\mathrm{d}\boldsymbol{x}$ is the partition function which is analytically intractable for high dimensional $\boldsymbol{x}$.

2.1 Tranformation

Any standard distribution for a random variable or vector $\boldsymbol{X} : \Omega \to \mathbb{R}^d$ with a continuous density $f_{\boldsymbol{X}}$ can be transformed into an EBM. For this, an analytical representation of the probability kernel $K_{\boldsymbol{X}}$ of the distribution, i.e. the un-normalised version of the density, is already sufficient, making it an excellent tool for Bayesian models where the normalisation constant of the posterior is typically analytically unavailable or intractable. Given the kernel we can set

$$U(\boldsymbol{x}) = -\log K_{\boldsymbol{X}}(\boldsymbol{x}),$$

which reproduces the desired density when substituted in the EBMs density form:

$$\begin{aligned} p(\boldsymbol{x}) &= \frac{1}{Z}\exp(-U(\boldsymbol{x})) \\ &= \frac{1}{\int \exp(-U(\boldsymbol{x}))\,\mathrm{d}\boldsymbol{x}}\exp(-U(\boldsymbol{x})) \\ &= \frac{1}{\int \exp(\log K_{\boldsymbol{X}}(\boldsymbol{x}))\,\mathrm{d}\boldsymbol{x}}\exp(\log K_{\boldsymbol{X}}(\boldsymbol{x})) \\ &= \frac{1}{\int K_{\boldsymbol{X}}(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x}} K_{\boldsymbol{X}}(\boldsymbol{x}) \\ &= f_{\boldsymbol{X}}(\boldsymbol{x}) \end{aligned}$$

We might not have an analytical representation for density or kernel but only a way of calculating the density value. For analytical consideration it is helpful to be able to still have a representation for the energy functional. We know that for a model with parameterised density $p_{\boldsymbol{\theta}}$ we have

$$\begin{aligned} -\log p_{\boldsymbol{\theta}}(\boldsymbol{x}) &= -\log \frac{K_{\boldsymbol{\theta}}(\boldsymbol{x})}{Z_{\boldsymbol{\theta}}} \\ &= -\log K_{\boldsymbol{\theta}}(\boldsymbol{x}) + \log Z_{\boldsymbol{\theta}} \end{aligned}$$

so we can represent the energy functional as

$$U_{\boldsymbol{\theta}}(\boldsymbol{x}) = -\log p_{\boldsymbol{\theta}}(\boldsymbol{x}) - \log Z_{\boldsymbol{\theta}}$$

2.2 Differentiating

For sampling and estimation we will have to differentiate the energy functional of a model by both the input $\boldsymbol{x}$ and by the parameter vector $\boldsymbol{\theta}$.
In general, for a function $f : \mathbb{R}^d \to \mathbb{R}$ we have by the chain rule that

$$\nabla_{\boldsymbol{x}}(-\log f(\boldsymbol{x})) = -\frac{1}{f(\boldsymbol{x})}\nabla_{\boldsymbol{x}} f(\boldsymbol{x}) = -\frac{\nabla_{\boldsymbol{x}} f(\boldsymbol{x})}{f(\boldsymbol{x})}$$

so for a model were we have the kernel analytically the derivative with respect to $\boldsymbol{x}$ is

$$\begin{aligned} \nabla_{\boldsymbol{x}} U_{\boldsymbol{\theta}}(\boldsymbol{x}) &= \nabla_{\boldsymbol{x}}(-\log K_{\boldsymbol{\theta}}(\boldsymbol{x})) \\ &= -\frac{\nabla_{\boldsymbol{x}} K_{\boldsymbol{\theta}}(\boldsymbol{x})}{K_{\boldsymbol{\theta}}(\boldsymbol{x})} \end{aligned}$$

and analogously for $\boldsymbol{\theta}$:

$$\nabla_{\boldsymbol{\theta}} U_{\boldsymbol{\theta}}(\boldsymbol{x}) = -\frac{\nabla_{\boldsymbol{\theta}} K_{\boldsymbol{\theta}}(\boldsymbol{x})}{K_{\boldsymbol{\theta}}(\boldsymbol{x})}.$$

In the other case where we don't have the kernel explicitly the calculation is similar with respect to $\boldsymbol{x}$ as the normalisation constant falls away

$$\begin{aligned}
\nabla_{\boldsymbol{x}} U_{\boldsymbol{\theta}}(\boldsymbol{x}) &= \nabla_{\boldsymbol{x}}(-\log p_{\boldsymbol{\theta}}(\boldsymbol{x}) - \log Z_{\boldsymbol{\theta}}) \\
&= -\nabla_{\boldsymbol{x}} \log p_{\boldsymbol{\theta}}(\boldsymbol{x}) - \nabla_{\boldsymbol{x}} \log Z_{\boldsymbol{\theta}} \\
&= -\frac{\nabla_{\boldsymbol{x}} p_{\boldsymbol{\theta}}(\boldsymbol{x})}{p_{\boldsymbol{\theta}}(\boldsymbol{x})}
\end{aligned}$$

but more difficult when differentiating with respect to $\boldsymbol{\theta}$ as $Z_{\boldsymbol{\theta}}$ is a function of it.

$$\begin{aligned}
\nabla_{\boldsymbol{\theta}} U_{\boldsymbol{\theta}}(\boldsymbol{x}) &= \nabla_{\boldsymbol{\theta}}(-\log p_{\boldsymbol{\theta}}(\boldsymbol{x}) - \log Z_{\boldsymbol{\theta}}) \\
&= -\nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\boldsymbol{x}) - \nabla_{\boldsymbol{\theta}} \log Z_{\boldsymbol{\theta}} \\
&= -\frac{\nabla_{\boldsymbol{\theta}} p_{\boldsymbol{\theta}}(\boldsymbol{x})}{p_{\boldsymbol{\theta}}(\boldsymbol{x})} - \frac{\nabla_{\boldsymbol{\theta}} Z_{\boldsymbol{\theta}}}{Z_{\boldsymbol{\theta}}}
\end{aligned}$$

3 Distributions

Let $X : \Omega \to \mathbb{R}$ or $\boldsymbol{X} : \Omega \to \mathbb{R}^d$ be a random variable or a random vector respectively, with the respective distribution.

3.1 The Exponential Family

A parameterised exponential family is a set of probability distributions whose probability density function p can be expressed in the form

$$p(\boldsymbol{x}|\boldsymbol{\theta}) = h(\boldsymbol{x}) \exp\left[\eta(\boldsymbol{\theta})^T \phi(\boldsymbol{x}) - A(\eta(\boldsymbol{\theta}))\right]$$

where

1. $\boldsymbol{\theta}$ is called *natural* or *canonical* parameters
2. $\phi(\boldsymbol{x})$ is the vector of *sufficient statistics*
3. η maps the parameters $\boldsymbol{\theta}$ to the canonical parameters
4. A is the *log-partition function* or *cumulant function*

We can rearrange the form of the exponential family to obtain an EBM version:

$$\begin{aligned}
p(\boldsymbol{x}|\boldsymbol{\theta}) &= h(\boldsymbol{x}) \exp\left[\eta(\boldsymbol{\theta})^T \phi(\boldsymbol{x}) - A(\eta(\boldsymbol{\theta}))\right] \\
&= \frac{1}{\exp\left(A(\eta(\boldsymbol{\theta}))\right)} \exp\left[\eta(\boldsymbol{\theta})^T \phi(\boldsymbol{x}) + \log(h(\boldsymbol{x}))\right]
\end{aligned}$$

and define the energy function of such a model as

$$U_{\boldsymbol{\theta}}(\boldsymbol{x}) = -\eta(\boldsymbol{\theta})^T \phi(\boldsymbol{x}) - \log(h(\boldsymbol{x})).$$

The gradient w.r.t. $\boldsymbol{x}$ is then given by

$$\nabla_{\boldsymbol{x}} U_{\boldsymbol{\theta}}(\boldsymbol{x}) = -\eta(\boldsymbol{\theta})^T J_\phi(\boldsymbol{x}) - \frac{\nabla_{\boldsymbol{x}} h(\boldsymbol{x})}{h(\boldsymbol{x})}$$

where J_ϕ is the Jacobian matrix of ϕ with entries $(J_\phi(\boldsymbol{x}))_{i,j} = \frac{\partial}{\partial x_j}\phi(\boldsymbol{x})_i$ corresponding to the partial derivatives of the components of $\phi(\boldsymbol{x})$. The gradient w.r.t. the parameter similarly becomes

$$\nabla_{\boldsymbol{\theta}} U_{\boldsymbol{\theta}}(\boldsymbol{x}) = -\phi(\boldsymbol{x})^T J_\eta(\boldsymbol{\theta})$$

3.2 Multivariate Normal Distribution

The multivariate normal distribution is used a lot in the process of this report, so we state some important properties explicitly here. For mean $\boldsymbol{\mu} \in \mathbb{R}^d$ and standard deviation $\Sigma \in \mathbb{R}^{d\times d}$ we say X follows a multivariate Gaussian or normal distribution $X \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma)$, if its probability density function is given by: **Probability Density Function (pdf):**

$$f_X(\boldsymbol{x}|\boldsymbol{\mu}, \Sigma) = \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T\Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right)$$

Kernel:

$$K_X(\boldsymbol{x}) = \exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T\Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right)$$

Energy Function:

$$\begin{aligned} U(\boldsymbol{x}) &= -\log(K_X(\boldsymbol{x})) \\ &= -\log(\exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T\Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right)) \\ &= \left(\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T\Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right) \end{aligned}$$

Gradient of Energy Function with respect to $\boldsymbol{x}$:

$$\begin{aligned} \nabla U(\boldsymbol{x}) &= \nabla\left(\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T\Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right) \\ &= \Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu}) \end{aligned}$$

Gradient of Energy Function with respect to $\boldsymbol{\mu}$:

$$\begin{aligned} \nabla_{\boldsymbol{\mu}} U(\boldsymbol{x}) &= \nabla_{\boldsymbol{\mu}}\left(\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T\Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right) \\ &= \Sigma^{-1}(\boldsymbol{\mu}-\boldsymbol{x}) \end{aligned}$$

The partial derivatives of the energy functional with respect to $\Sigma_{i,j}$ however are not directly obtainable as there is no general form of Σ^{-1} available. Thus it might be preferable for experiments to use a diagonal covariance matrix where this derivative could be directly stated.

3.3 Gaussian Mixture Distribution (GMD)

Gaussian mixture distributions make for simple but sufficiently complex distribution for our parameter estimation testing process. A GMD is a random vector $\boldsymbol{X} : \Omega \to \mathbb{R}^d$, whose density is the weighted sum of the densities $f_i : \mathbb{R}^d \to \mathbb{R}$, of m independent (?) Gaussian random vectors $\boldsymbol{Z}_i \sim \mathcal{N}(\boldsymbol{\mu}_i, \Sigma_i)$ with corresponding weights $w_i \in (0, 1)$:

$$p_{\boldsymbol{\theta}}(\boldsymbol{x}) := \sum_{i=1}^{m} w_i f_i(\boldsymbol{x})$$

where we aggregate the weights, means and covariances in the parameter $\boldsymbol{\theta}$ for convenience. For each $i \in [m]$ we can split the density f_i in two parts, namely the kernel which, we denote as K_i, and the normalisation constant Z_i, i.e.

$$f_i(\boldsymbol{x}) = \underbrace{\frac{1}{\sqrt{(2\pi)^d |\Sigma_i|}}}_{=:Z_i} \underbrace{\exp\left(-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu}_i)^T \Sigma_i^{-1} (\boldsymbol{x} - \boldsymbol{\mu}_i)\right)}_{=:K_i(\boldsymbol{x})}$$

and rewrite the density with these as

$$\begin{aligned}
p_{\boldsymbol{\theta}}(\boldsymbol{x}) &= \sum_{i=1}^{m} w_i \frac{K_i(\boldsymbol{x})}{Z_i} \\
&= \sum_{i=1}^{m} \frac{w_i (\prod_{j \neq i} Z_j) K_i(\boldsymbol{x})}{\prod_{j=1}^{m} Z_j} \\
&= \frac{\sum_{i=1}^{m} w_i (\prod_{j \neq i} Z_j) K_i(\boldsymbol{x})}{\prod_{j=1}^{m} Z_j}
\end{aligned}$$

By defining

$$K_{\boldsymbol{\theta}}(\boldsymbol{x}) = \sum_{i=1}^{m} w_i (\prod_{j \neq i} Z_j) K_i(\boldsymbol{x})$$

we obtain this expression for the energy $U_{\boldsymbol{\theta}}$:

$$\begin{aligned}
U_{\boldsymbol{\theta}}(\boldsymbol{x}) &= -\log(K_{\boldsymbol{\theta}}(\boldsymbol{x})) \\
&= \log\left(\frac{1}{\sum_{i=1}^{m} w_i (\prod_{j \neq i} Z_j) K_i(\boldsymbol{x})}\right)
\end{aligned}$$

Here neither the gradient of $U_{\boldsymbol{\theta}}$ w.r.t. the inputs $\boldsymbol{x}$ nor w.r.t. to the parameters $\boldsymbol{\theta}$ is simple to calculate, and hence we will leverage the automatic differentiation capacity of pytorch for these.

4 Metropolis-Hastings

Let $\tilde{p} : \mathbb{R}^d \to \mathbb{R}$ be the unnormalised probability density function, of the density

$$p(\boldsymbol{x}) := \frac{\tilde{p}(\boldsymbol{x})}{\int_{\Theta} \tilde{p}(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x}}$$

with a potentially intractable normalisation constant $\int_{\Theta} \tilde{p}(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x}$.

Metropolis-Hastings

Initialise $\boldsymbol{x}_0$

for $k \leq n$ **do**

  sample $\hat{\boldsymbol{x}} \sim q(\hat{\boldsymbol{x}}|\boldsymbol{x}_k)$

  compute $\alpha(\boldsymbol{x}_k, \hat{\boldsymbol{x}}) = \min\{1, \frac{\tilde{p}(\hat{\boldsymbol{x}})q(\boldsymbol{x}_k|\hat{\boldsymbol{x}})}{\tilde{p}(\boldsymbol{x}_k)q(\hat{\boldsymbol{x}}|\boldsymbol{x}_k)}\}$

  sample $u \sim \mathcal{U}(0, 1)$

  if $u \leq \alpha(\boldsymbol{x}_k, \hat{\boldsymbol{x}})$ **then**

    $\boldsymbol{x}_{k+1} := \hat{\boldsymbol{x}}$

  else

    $\boldsymbol{x}_{k+1} := \boldsymbol{x}_k$

We can use the unnormalised $\tilde{p}(\cdot)$ instead of $p(\cdot)$ as the normalisation constant cancels in the fraction. A typical proposal distribution driving the Markov chain is $q(\cdot|\boldsymbol{x}) \sim \mathcal{N}(\cdot|\boldsymbol{x}, \boldsymbol{\Sigma})$, mimicking a random walk in the parameter space. To achieve high acceptance rates one can choose smaller proposed transitions. But this approach increases the traversal time and can become inefficient for high dimensional parameter spaces. This can imply row rates of acceptance, resulting in a highly correlated Markov chain and/or high mixing times. The mixing time refers to the time it takes until the chain reaches the target distribution, while the correlation comes from the chain remaining in the same states when the transition is rejected.

5 Langevin Dynamics

5.1 Langevin Equation

The Langevin equation describes the evolution of a particle's position in a dissipative medium and is given by:

$$m\frac{d^2x}{dt^2} = -\gamma\frac{dx}{dt} + \sqrt{2kT\gamma}\eta(t)$$

where m is the mass, γ is the friction coefficient, x is the position of the particle, T is the temperature and $\eta(t)$ is a white noise term. The Langevin equation combines deterministic motion with a stochastic term to model the impact of random collisions with surrounding particles in a medium. It's widely used in statistical mechanics to describe the behaviour of particles under the influence of both deterministic and random forces.

5.2 Langevin Diffusion

The Langevin diffusion process is a stochastic differential equation (SDE) that represents the continuous-time evolution of a particle's position and is given by:

$$d\boldsymbol{X}_t = -\frac{1}{\gamma}\nabla U(\boldsymbol{X}_t)dt + \sqrt{\frac{2kT}{\gamma}}d\boldsymbol{B}_t$$

where $U(x)$ is the potential energy at position x and $\boldsymbol{B}$ is a Brownian motion. Langevin diffusion describes the random motion of a particle in a potential field, incorporating the effects of both deterministic drift and stochastic noise.

5.3 Invariant Measure via Fokker-Planck Equation

Consider a simpler form of the Langevin diffusion SDE:

$$\mathrm{d}\boldsymbol{X}_t = -\nabla U(\boldsymbol{X}_t)\,\mathrm{d}t + \sqrt{2}\,\mathrm{d}\boldsymbol{B}_t$$

where $\gamma = 1$ and $T = k = 1$. The Fokker-Planck Equation for the density of the solution $p(\boldsymbol{x}, t)$ reads

$$\begin{aligned}
\partial_t p(\boldsymbol{x}, t) &= \sum_i \partial_{x_i}\left[\partial_{x_i} U(\boldsymbol{x}) p(\boldsymbol{x}, t)\right] + \sum_i \sum_j \partial_{x_i}\partial_{x_j}\delta_{i,j} p(\boldsymbol{x}, t) \\
&= \sum_i \partial_{x_i}\left[\partial_{x_i} U(\boldsymbol{x}) p(\boldsymbol{x}, t)\right] + \sum_i \partial^2_{x_i} p(\boldsymbol{x}, t) \\
&= \nabla \cdot \left(\nabla U(\boldsymbol{x}) p(\boldsymbol{x}, t) + \nabla p(\boldsymbol{x}, t)\right)
\end{aligned}$$

We can use this equation to show that the distribution with density

$$\pi(\boldsymbol{x}) = \frac{1}{\int \exp(-U(\boldsymbol{y}))d\boldsymbol{y}} \exp(-U(\boldsymbol{x})) =: \frac{1}{Z}\exp(-U(\boldsymbol{x}))$$

is an invariant measure of the process defined by this SDE. To see this, consider that the gradient of π with respect to $\boldsymbol{x}$ is given by:

$$\nabla\pi(\boldsymbol{x}) = \frac{1}{Z}\nabla\exp(-U(\boldsymbol{x})) = -\frac{1}{Z}\nabla U(\boldsymbol{x})\exp(-U(\boldsymbol{x})) = -\nabla U(\boldsymbol{x})\pi(\boldsymbol{x})$$

So when we replace $p(\boldsymbol{x}, t)$ on the r.h.s. of the Fokker-Planck equation with $\pi(\boldsymbol{x})$, we have that

$$\begin{aligned}
\partial_t p(\boldsymbol{x}, t) &= \nabla \cdot (\nabla U(\boldsymbol{x})\pi(\boldsymbol{x}) + \nabla\pi(\boldsymbol{x})) \\
&= \nabla \cdot (\nabla U(\boldsymbol{x})\pi(\boldsymbol{x}) - \nabla U(\boldsymbol{x})\pi(\boldsymbol{x})) \\
&= 0
\end{aligned}$$

Thus, if the Langevin diffusion process attains the measure given by $\pi(\boldsymbol{x})$ it will not change anymore and hence $\pi(\boldsymbol{x})$ is an invariant measure of the process. This means for one that the Langevin diffusion process can be made to converge to a desired density of the exponential family. On the other hand one can use it to sample from any unnormalised density $\tilde{p}(\boldsymbol{x})$ as taking $-U(\boldsymbol{x}) := \log\tilde{p}(\boldsymbol{x})$ means the stationary density of the Langevin diffusion will be

$$\begin{aligned}
\pi(\boldsymbol{x}) &= \frac{\exp(-U(\boldsymbol{x}))}{\int \exp(-U(\boldsymbol{y}))d\boldsymbol{y}} \\
&= \frac{\tilde{p}(\boldsymbol{x})}{\int \tilde{p}(\boldsymbol{y})d\boldsymbol{y}} \\
&= p(\boldsymbol{x})
\end{aligned}$$

i.e. the density of the stationary distribution of this Langevin diffusion process corresponds to the normalised version of $\tilde{p}$.

5.4 Unadjusted Langevin Algorithm (ULA)

The problem in general is that we do not have a closed form solution for the Langevin diffusion SDE, and cannot directly sample from the process it defines. The straightforward approach then is to simulate the process by discretising it and sampling from the discrete approximation instead.

Discretising the SDE with the Euler-Maruyama scheme is the basis for the unadjusted Langevin Algorithm. For the SDE of the Langevin diffusion process this discretisation produces

$$\hat{\boldsymbol{X}}_{t_{i+1}} = \hat{\boldsymbol{X}}_{t_i} - \nabla U(\hat{\boldsymbol{X}}_{t_i})[t_{i+1} - t_i] + \sqrt{2(t_{i+1} - t_i)}\boldsymbol{Z}_i$$

where the $\boldsymbol{Z}_i$ are i.i.d. normal distributed, i.e. $\boldsymbol{Z}_i \sim \mathcal{N}(0, \boldsymbol{I})$. Let's modify this equation to reflect an equidistant grid of time points with step size ε and let $\boldsymbol{X}_n := \hat{\boldsymbol{X}}_{t_n}$ be the discretised approximation of $\boldsymbol{X}$ w.r.t. this grid at step n. We then have:

$$\boldsymbol{X}_{n+1} = \boldsymbol{X}_n - \varepsilon\nabla U(\boldsymbol{X}_n) + \sqrt{2\varepsilon}\boldsymbol{Z}_n$$

Which gives us a chain of connected random variables. So we can simulate realisations of this discrete process with this simple algorithm

---

Unadjusted Langevin Algorithm

Initialise $\boldsymbol{x}_0$
for $n \leq N$ **do**
  sample $\boldsymbol{Z}_n \sim \mathcal{N}(0, \boldsymbol{I})$
  $\boldsymbol{x}_{n+1} := \boldsymbol{x}_n - \varepsilon\nabla U(\boldsymbol{x}_n) + \sqrt{2\varepsilon}\boldsymbol{Z}_n$

This scheme incurs a discretisation/integration error however that could change the stationary distribution. This problem gives rise to the Metropolis adjusted Langevin algorithm that introduces an acceptance probability to ensure convergence to the desired stationary distribution.

5.5 Metropolis Adjusted Langevin Algorithm (MALA)

The MALA method is a Metropolis-Hastings algorithm where the proposal density is based on the Langevin diffusion. We can now adapt our point of view and consider the next point $\boldsymbol{X}_{n+1}$ as the suggestion of our proposal density. For this we take this not as a discrete process rather think of a fixed vector $\boldsymbol{x}_n$ as a concrete realisation of $\boldsymbol{X}_n$. So for a fixed specific step the random vector $\hat{\boldsymbol{X}}$ is given by

$$\hat{\boldsymbol{X}} = \boldsymbol{x}_n - \varepsilon\nabla U(\boldsymbol{x}_n) + \sqrt{2\varepsilon}\boldsymbol{Z}_n$$

Then the law of $\hat{\boldsymbol{X}}$ inherits the normal distribution of $\boldsymbol{Z}_n$ with mean

$$\begin{aligned}\mathbb{E}\left[\hat{\boldsymbol{X}}\right] &= \mathbb{E}\left[\boldsymbol{x}_n - \varepsilon\nabla U(\boldsymbol{x}_n) + \sqrt{2\varepsilon}\boldsymbol{Z}_n\right] \\ &= \boldsymbol{x}_n - \varepsilon\nabla U(\boldsymbol{x}_n) + \sqrt{2\varepsilon}\mathbb{E}\left[\boldsymbol{Z}_n\right] \\ &= \boldsymbol{x}_n - \varepsilon\nabla U(\boldsymbol{x}_n).\end{aligned}$$

The difference between $\hat{\boldsymbol{X}}$ and its expectation is thus

$$\begin{aligned}\hat{\boldsymbol{X}} - \mathbb{E}\left[\hat{\boldsymbol{X}}\right] &= \left(\boldsymbol{x}_n - \varepsilon\nabla U(\boldsymbol{x}_n) + \sqrt{2\varepsilon}\boldsymbol{Z}_n\right) - (\boldsymbol{x}_n - \varepsilon\nabla U(\boldsymbol{x}_n)) \\ &= \sqrt{2\varepsilon}\boldsymbol{Z}_n\end{aligned}$$

and with this term we can directly calculate the variance

$$\begin{aligned}\mathrm{Var}(\hat{\boldsymbol{X}}) &= \mathbb{E}\left[(\hat{\boldsymbol{X}} - \mathbb{E}\left[\hat{\boldsymbol{X}}\right])^T(\hat{\boldsymbol{X}} - \mathbb{E}\left[\hat{\boldsymbol{X}}\right])\right] \\ &= \mathbb{E}\left[(\sqrt{2\varepsilon}\boldsymbol{Z}_n)^T(\sqrt{2\varepsilon}\boldsymbol{Z}_n)\right] \\ &= 2\varepsilon\mathbb{E}\left[\boldsymbol{Z}_n^T\boldsymbol{Z}_n\right] \\ &= 2\varepsilon\boldsymbol{I}.\end{aligned}$$

With this we can define a Metropolis-Hastings proposal density

$$q(\hat{\boldsymbol{x}}|\boldsymbol{x}_n) = \mathcal{N}(\hat{\boldsymbol{x}}|\boldsymbol{x}_n - \varepsilon\nabla U(\boldsymbol{x}_n); 2\varepsilon\boldsymbol{I})$$

and state the complete algorithm:

Metropolis Adjusted Langevin Algorithm

Initialise $\boldsymbol{x}_0$
for $n \leq N$ **do**
    sample $\hat{\boldsymbol{x}} \sim \mathcal{N}(\boldsymbol{x}_n - \varepsilon\nabla U(\boldsymbol{x}_n), 2\varepsilon\boldsymbol{I}))$
    compute $\alpha(\boldsymbol{x}_n, \hat{\boldsymbol{x}}) = \min\left\{1, \frac{p(\hat{\boldsymbol{x}})q(\boldsymbol{x}_n|\hat{\boldsymbol{x}})}{p(\boldsymbol{x}_n)q(\hat{\boldsymbol{x}}|\boldsymbol{x}_n)}\right\}$
    sample $u \sim \mathcal{U}(0, 1)$
    if $u \leq \alpha(\boldsymbol{x}_n, \hat{\boldsymbol{x}})$ **then**
        $\boldsymbol{x}_{n+1} := \hat{\boldsymbol{x}}$
    else
        $\boldsymbol{x}_{n+1} := \boldsymbol{x}_n$

In order to implement the algorithm efficiently we would like to get a simplified expression for the acceptance probability

$$\alpha(\boldsymbol{x}_n, \hat{\boldsymbol{x}}) = \min\left\{1, \frac{p(\hat{\boldsymbol{x}})q(\boldsymbol{x}_n|\hat{\boldsymbol{x}})}{p(\boldsymbol{x}_n)q(\hat{\boldsymbol{x}}|\boldsymbol{x}_n)}\right\}$$

For the ratio of the target density we get

$$\frac{p(\hat{\boldsymbol{x}})}{p(\boldsymbol{x}_n)} = \frac{\exp(-U(\hat{\boldsymbol{x}}))}{\exp(-U(\boldsymbol{x}_n))} = \exp(U(\boldsymbol{x}_n) - U(\hat{\boldsymbol{x}}))$$

as the normalisation constant cancels in the ratio.
When substituting the density for the multivariate normal in the proposal density we get for general $\boldsymbol{x}, \boldsymbol{y}$,

$$q(\boldsymbol{x}|\boldsymbol{y}) = \frac{\exp\left(-\frac{1}{4\varepsilon}\left\|\boldsymbol{x} - \boldsymbol{y} + \varepsilon\nabla U(\boldsymbol{y})\right\|_2^2\right)}{\sqrt{(4\pi\varepsilon)^d}}$$

So with $\hat{\boldsymbol{x}}, \boldsymbol{x}_n$, we can write the ratio as

$$\begin{aligned}
\frac{q(\boldsymbol{x}_n|\hat{\boldsymbol{x}})}{q(\hat{\boldsymbol{x}}|\boldsymbol{x}_n)} &= \frac{\frac{\exp\left(-\frac{1}{4\varepsilon}\|\boldsymbol{x}_n - \hat{\boldsymbol{x}} + \varepsilon\nabla U(\hat{\boldsymbol{x}})\|_2^2\right)}{\sqrt{(4\pi\varepsilon)^d}}}{\frac{\exp\left(-\frac{1}{4\varepsilon}\|\hat{\boldsymbol{x}} - \boldsymbol{x}_n + \varepsilon\nabla U(\boldsymbol{x}_n)\|_2^2\right)}{\sqrt{(4\pi\varepsilon)^d}}} \\
&= \frac{\exp\left(-\frac{1}{4\varepsilon}\left\|\boldsymbol{x}_n - \hat{\boldsymbol{x}} + \varepsilon\nabla U(\hat{\boldsymbol{x}})\right\|_2^2\right)}{\exp\left(-\frac{1}{4\varepsilon}\left\|\hat{\boldsymbol{x}} - \boldsymbol{x}_n + \varepsilon\nabla U(\boldsymbol{x}_n)\right\|_2^2\right)} \\
&= \exp\left[\frac{1}{4\varepsilon}\left\|\hat{\boldsymbol{x}} - \boldsymbol{x}_n + \varepsilon\nabla U(\boldsymbol{x}_n)\right\|_2^2 - \frac{1}{4\varepsilon}\left\|\boldsymbol{x}_n - \hat{\boldsymbol{x}} + \varepsilon\nabla U(\hat{\boldsymbol{x}})\right\|_2^2\right] \\
&= \exp\left[-\frac{1}{4\varepsilon}\left(\left\|\boldsymbol{x}_n - \hat{\boldsymbol{x}} + \varepsilon\nabla U(\hat{\boldsymbol{x}})\right\|_2^2 - \left\|\hat{\boldsymbol{x}} - \boldsymbol{x}_n + \varepsilon\nabla U(\boldsymbol{x}_n)\right\|_2^2\right)\right]
\end{aligned}$$

Together we get the acceptance probability

$$\alpha(\boldsymbol{x}_n, \hat{\boldsymbol{x}}) = \min\left\{1, \exp\left(U(\boldsymbol{x}_n) - U(\hat{\boldsymbol{x}}) - \frac{1}{4\varepsilon}\left(\left\|\boldsymbol{x}_n - \hat{\boldsymbol{x}} + \varepsilon\nabla U(\hat{\boldsymbol{x}})\right\|_2^2 - \left\|\hat{\boldsymbol{x}} - \boldsymbol{x}_n + \varepsilon\nabla U(\boldsymbol{x}_n)\right\|_2^2\right)\right)\right\}$$

5.6 Sampling Accuracy

It is important to note here that in practice we can not run the corresponding algorithm for an indefinite amount of time. A sampler based on MALA, ULA or HMC is used for generating training samples from the parametric model.

Typically one either just takes the first m samples in the chains, or lets the sampler run for some K iterations before taking the subsequent samples. That K is often called burn-in period and can improve the quality of the samples at the cost of more computation time.

When cutting a samplers chains off we incur two fundamental errors. The first is the error due to the discretisation of the process that is using $\hat{\boldsymbol{X}}_t$ instead of the true $\boldsymbol{X}_t$. It depends on the discretisation step ε and is meant to be counteracted by the Metropolis Hastings acceptance step in MALA. We are interested in strong convergence here and we know that the convergence order of the Euler-Maruyama discretisation is proportional to $\sqrt{\varepsilon}$. For ULA this is significantly more important as the quality of the samples suffers directly, while for MALA a poorly chosen ε means that less new samples will be accepted which can either impact quality or computational effort.

The second error is due to incomplete convergence, i.e. even if we could sample from $\boldsymbol{X}_t$ directly, we only have a guarantee that $\boldsymbol{X}$ converges to the target distribution in the limit, but might not resemble it closely enough at a specific t. One could run convergence diagnostics like the Gelman-Rubin test to assess the convergence status. That would introduce computational overhead however and complicate the code. Another option is to set a specific burn-in period and hope for sufficient convergence.

6 MCMC using Hamiltonian Dynamics

The idea to reconsider a distributions density as an energy landscape is the foundation of not only ULA and MALA but also of Hamiltonian Monte Carlo (HMC). HMC is a much more general concept however, that, in addition to the potential energy function U that characterises an EBM, introduces momentum variables to assign both potential as well as kinetic energy to a state via the Hamiltonian. The combination of these two energy functionals is then subjected to Hamiltonian dynamics to generate a trajectory to the next proposal state. MALA is in principle a special case of HMC where the trajectory to generate a new proposal consists of a single step.

6.1 Hamiltonian Dynamics

Hamiltonian dynamics is a framework used in classical mechanics to describe the evolution of physical systems. It introduces the concept of a Hamiltonian function, denoted as $H(\boldsymbol{x}, \boldsymbol{p})$,

$$H(\boldsymbol{x}, \boldsymbol{p}) = U(\boldsymbol{x}) + K(\boldsymbol{p})$$

where $\boldsymbol{x}$ represents the position vector and $\boldsymbol{p}$ the momentum vector and U and K are called the potential and kinetic energy respectively. Hamiltonian dynamics are governed by the equations:

$$\frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}t} = \frac{\partial H}{\partial \boldsymbol{p}}$$
$$\frac{\mathrm{d}\boldsymbol{p}}{\mathrm{d}t} = -\frac{\partial H}{\partial \boldsymbol{x}}$$

For an EBM Model the potential energy function coincides with U, and for the kernel of any other density we can do the familiar transformation $U(\boldsymbol{x}) = -\log(\tilde{f}(x))$. The Hamiltonian in its turn can be used as the energy function in a new EBM to define a joint density

$$P(\boldsymbol{x}, \boldsymbol{p}) = \frac{1}{Z}\exp(-H(\boldsymbol{x}, \boldsymbol{p})) = \frac{1}{Z}\exp(-U(\boldsymbol{x}) - K(\boldsymbol{p}))$$

6.2 Hamiltonian Monte Carlo (HMC)

HMC is a Metropolis-Hastings sampling algorithm that employs Hamiltonian dynamics to generate new proposal states, and in this way guide the exploration of the state space efficiently. The simulated dynamics allow the algorithm to explore distant regions in a more coherent and directed manner compared to random-walk-based methods. Unlike traditional Metropolis Hastings algorithms, that often sample the proposed next state using a multivariate-normal step directly, HMC samples a momentum and subsequently uses the deterministic dynamics to simulate a trajectory from the current position to the next proposal state. Unlike in Langevin Dynamics this trajectory can consist of multiple steps which can boost convergence speed.

A key component is to negate the momentum variables at the end of one trajectory simulation. This guarantees that the proposal distribution q is symmetric which makes it vanish in the Metropolis-Hastings acceptance probability

$$\begin{aligned}\alpha(\boldsymbol{x},\boldsymbol{p};\hat{\boldsymbol{x}},\hat{\boldsymbol{p}}) &= \min\left(1, \frac{P(\hat{\boldsymbol{x}},\hat{\boldsymbol{p}})q(\boldsymbol{x},\boldsymbol{p}|\hat{\boldsymbol{x}},\hat{\boldsymbol{p}})}{P(\boldsymbol{x},\boldsymbol{p})}q(\hat{\boldsymbol{x}},\hat{\boldsymbol{p}}|\boldsymbol{x},\boldsymbol{p})\right)\\ &= \min\left(1, \frac{P(\hat{\boldsymbol{x}},\hat{\boldsymbol{p}})}{P(\boldsymbol{x},\boldsymbol{p})}\right)\\ &= \min\left(1, \frac{P(\hat{\boldsymbol{x}},\hat{\boldsymbol{p}})}{P(\boldsymbol{x},\boldsymbol{p})}\right)\end{aligned}$$

So that it can be expressed purely in terms of the respective Hamiltonians

$$\begin{aligned}\alpha(\boldsymbol{x},\boldsymbol{p};\hat{\boldsymbol{x}},\hat{\boldsymbol{p}}) &= \min\left(1, \frac{P(\hat{\boldsymbol{x}},\hat{\boldsymbol{p}})}{P(\boldsymbol{x},\boldsymbol{p})}\right)\\ &= \min\left(1, \frac{\exp(-H(\hat{\boldsymbol{x}},\hat{\boldsymbol{p}}))}{\exp(-H(\boldsymbol{x},\boldsymbol{p}))}\right)\\ &= \min\left(1, \exp(H(\boldsymbol{x},\boldsymbol{p}) - H(\hat{\boldsymbol{x}},\hat{\boldsymbol{p}}))\right)\end{aligned}$$

HMC's ability to use gradient information and simulate deterministic dynamics makes it a powerful tool for efficient exploration of complex, high-dimensional parameter spaces.

Set with the parameters

1. Desired number of samples: N
2. Number of leapfrog steps: L
3. Step size for leapfrog integration: ε
4. Mass matrix for kinetic energy: $\boldsymbol{M}$

and with concrete kinetic energy function $K(\boldsymbol{p}) = \frac{\boldsymbol{p}^T M \boldsymbol{p}}{2}$

Hamiltonian Monte Carlo Algorithm $(N, L, \varepsilon, \boldsymbol{M})$

Initialise: $\boldsymbol{x}_0$
$\boldsymbol{p}_0 \sim \mathcal{N}(0, \boldsymbol{M})$
for $i \leq N$ **do**
  $\hat{\boldsymbol{x}} = \boldsymbol{x}_i$
  sample Momentum: $\hat{\boldsymbol{p}} \sim \mathcal{N}(0, \boldsymbol{M})$
  $\hat{\boldsymbol{p}} = \hat{\boldsymbol{p}} - \varepsilon \frac{\nabla U(\hat{\boldsymbol{x}})}{2}$
  leapfrog Integration:
  for $k \leq L - 1$ **do**
    $\hat{\boldsymbol{x}} = \hat{\boldsymbol{x}} + \varepsilon \hat{\boldsymbol{p}}$
    $\hat{\boldsymbol{p}} = \hat{\boldsymbol{p}} - \varepsilon \nabla U(\hat{\boldsymbol{x}})$
  $\hat{\boldsymbol{x}} = \hat{\boldsymbol{x}} + \varepsilon \hat{\boldsymbol{p}}$
  $\hat{\boldsymbol{p}} = \hat{\boldsymbol{p}} - \varepsilon \frac{\nabla U(\hat{\boldsymbol{x}})}{2}$
  negate Momentum: $\hat{\boldsymbol{p}} = -\hat{\boldsymbol{p}}$
  evaluate Hamiltonian for the current point: $H(\boldsymbol{x}_i, \boldsymbol{p}_i) = U(\boldsymbol{x}_i) + K(\boldsymbol{p}_i)$
  evaluate Hamiltonian for the proposed point: $H(\hat{\boldsymbol{x}}, \hat{\boldsymbol{p}}) = U(\hat{\boldsymbol{x}}) + K(\hat{\boldsymbol{p}})$
  compute $\alpha\left((\boldsymbol{x}_i, \boldsymbol{p}_i), (\hat{\boldsymbol{x}}, \hat{\boldsymbol{p}})\right) = \min\left\{1, \exp(H(\boldsymbol{x}_i, \boldsymbol{p}_i) - H(\hat{\boldsymbol{x}}, \hat{\boldsymbol{p}}))\right\}$
  sample $u \sim \mathcal{U}(0, 1)$
  if $u \leq \alpha\left((\boldsymbol{x}_i, \boldsymbol{p}_i), (\hat{\boldsymbol{x}}, \hat{\boldsymbol{p}})\right)$ **then**
    $\boldsymbol{x}_{i+1} := \hat{\boldsymbol{x}}$
  else
    $\boldsymbol{x}_{i+1} := \boldsymbol{x}_i$

7 Maximum Likelihood Estimation (MLE)

In principle an EBM can be learned with MLE where for a dataset $(\boldsymbol{x}_i)_{i=1}^n$ the normalised log-likelihood function is given as

$$\frac{1}{n} \log \mathcal{L}(\boldsymbol{\theta}) = \frac{1}{n} \log \prod_{i=1}^{n} p_{\boldsymbol{\theta}}(\boldsymbol{x}_i) = \frac{1}{n} \sum_{i=1}^{n} \log p_{\boldsymbol{\theta}}(\boldsymbol{x}_i) \doteq \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}}[\log p_{\boldsymbol{\theta}}(\boldsymbol{x})]$$

where the gradient with respect to $\boldsymbol{\theta}$ is

$$\begin{aligned}
\frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\theta}} \frac{1}{n} \log \mathcal{L}(\boldsymbol{\theta}) &= \frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\theta}} \left[\frac{1}{n} \sum_{i=1}^{n} \log p_{\boldsymbol{\theta}}(\boldsymbol{x}_i) \right] \\
&= \frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\theta}} \left[\frac{1}{n} \sum_{i=1}^{n} \log \left(\frac{1}{Z_{\boldsymbol{\theta}}} \exp(-U_{\boldsymbol{\theta}}(\boldsymbol{x}_i)) \right) \right] \\
&= \frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\theta}} \left[-\frac{1}{n} \sum_{i=1}^{n} U_{\boldsymbol{\theta}}(\boldsymbol{x}_i) - \frac{1}{n} \sum_{i=1}^{n} \log \left(Z_{\boldsymbol{\theta}} \right) \right] \\
&= \frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\theta}} \left[-\frac{1}{n} \sum_{i=1}^{n} U_{\boldsymbol{\theta}}(\boldsymbol{x}_i) - \log \left(Z_{\boldsymbol{\theta}} \right) \right] \\
&= \frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\theta}} \left[-\frac{1}{n} \sum_{i=1}^{n} U_{\boldsymbol{\theta}}(\boldsymbol{x}_i) - \log \left(\int \exp(U_{\boldsymbol{\theta}}(\boldsymbol{x})) d\boldsymbol{x} \right) \right] \\
&= -\frac{1}{n} \sum_{i=1}^{n} \frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\theta}} U_{\boldsymbol{\theta}}(\boldsymbol{x}_i) - \frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\theta}} \log \left(\int \exp(U_{\boldsymbol{\theta}}(\boldsymbol{x})) d\boldsymbol{x} \right)
\end{aligned}$$

where by the chain rule we get that

$$\frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\theta}} \log \left(\int \exp(-U_{\boldsymbol{\theta}}(\boldsymbol{x}))d\boldsymbol{x} \right) = \frac{\frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\theta}} \int \exp(-U_{\boldsymbol{\theta}}(\boldsymbol{x}))d\boldsymbol{x}}{Z_{\boldsymbol{\theta}}}$$

and assuming $\exp(-U_{\boldsymbol{\theta}}(\boldsymbol{x})$ is differentiable in $\boldsymbol{\theta}$ for all $\boldsymbol{x}$ and is integrable in $\boldsymbol{x}$ for all $\boldsymbol{\theta}$, then

$$\begin{aligned}
\frac{\frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\theta}} \int \exp(-U_{\boldsymbol{\theta}}(\boldsymbol{x}))d\boldsymbol{x}}{Z_{\boldsymbol{\theta}}} &= \frac{\int \frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\theta}} \exp(-U_{\boldsymbol{\theta}}(\boldsymbol{x}))d\boldsymbol{x}}{Z_{\boldsymbol{\theta}}} \\
&= -\frac{\int \frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\theta}} U_{\boldsymbol{\theta}}(\boldsymbol{x}) \exp(-U_{\boldsymbol{\theta}}(\boldsymbol{x}))d\boldsymbol{x}}{Z_{\boldsymbol{\theta}}} \\
&= -\int \frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\theta}} U_{\boldsymbol{\theta}}(\boldsymbol{x}) \frac{\exp(-U_{\boldsymbol{\theta}}(\boldsymbol{x}))}{Z_{\boldsymbol{\theta}}} d\boldsymbol{x} \\
&= -\mathbb{E}_{\boldsymbol{x}\sim p_{\boldsymbol{\theta}}} \left[\frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\theta}} U_{\boldsymbol{\theta}}(\boldsymbol{x}) \right]
\end{aligned}$$

The term $\frac{1}{n}\sum_{i=1}^{n} \frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\theta}} U_{\boldsymbol{\theta}}(\boldsymbol{x}_i)$ is a MC estimate for the expectation of the parameter gradient with respect to the data, i.e.

$$-\frac{1}{n}\sum_{i=1}^{n} \frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\theta}} U_{\boldsymbol{\theta}}(\boldsymbol{x}_i) \doteq -\mathbb{E}_{\boldsymbol{x}\sim p_{\mathrm{data}}} \left[\frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\theta}} U_{\boldsymbol{\theta}}(\boldsymbol{x}) \right]$$

so the final expression we get for $\frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\theta}} \log \mathcal{L}(\boldsymbol{\theta})$ is

$$\frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\theta}} \frac{1}{n} \log \mathcal{L}(\boldsymbol{\theta}) = \mathbb{E}_{\boldsymbol{x}\sim p_{\boldsymbol{\theta}}} \left[\frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\theta}} U_{\boldsymbol{\theta}}(\boldsymbol{x}) \right] - \mathbb{E}_{\boldsymbol{x}\sim p_{\mathrm{data}}} \left[\frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\theta}} U_{\boldsymbol{\theta}}(\boldsymbol{x}) \right].$$

For training we use the negative log likelihood (NLL) as a loss so the sign reverses and hence the gradient for our loss is

$$-\frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\theta}} \frac{1}{n} \log \mathcal{L}(\boldsymbol{\theta}) = \mathbb{E}_{\boldsymbol{x}\sim p_{\mathrm{data}}} \left[\frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\theta}} U_{\boldsymbol{\theta}}(\boldsymbol{x}) \right] - \mathbb{E}_{\boldsymbol{x}\sim p_{\boldsymbol{\theta}}} \left[\frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\theta}} U_{\boldsymbol{\theta}}(\boldsymbol{x}) \right].$$

In order to approximate this gradient we can use a MC estimate with samples from the respective distributions, using our dataset for the first, and generated samples from our models for the second expectation. In order for this estimation strategy to work and to apply the samplers introduced earlier we need to have a way to calculate the energy functional U and have it be differentiable, both with respect to the inputs $\boldsymbol{x}$ and with respect to the model parameters $\boldsymbol{\theta}$.

8 Recovery Likelihood

The idea of recovery likelihood is to create perturbed samples $\tilde{\boldsymbol{x}} = \boldsymbol{x} + \sigma\varepsilon$ with $\varepsilon \sim \mathcal{N}(0, I)$, from our original dataset and then use the conditional distribution

$$\begin{aligned}
p_{\boldsymbol{\theta}}(\boldsymbol{x}|\tilde{\boldsymbol{x}}) &= \frac{p(\tilde{\boldsymbol{x}}|\boldsymbol{x})p_{\boldsymbol{\theta}}(\boldsymbol{x})}{p(\tilde{\boldsymbol{x}})} \\
&= \frac{\exp(-\frac{\|\tilde{\boldsymbol{x}}-\boldsymbol{x}\|_2^2}{2\sigma^2}) \exp(-U_{\boldsymbol{\theta}}(\boldsymbol{x}))}{Z_{\boldsymbol{\theta}}(2\pi\sigma^2)^{\frac{n}{2}} p(\tilde{\boldsymbol{x}})} \\
&= \frac{\exp(-U_{\boldsymbol{\theta}}(\boldsymbol{x}) - \frac{\|\tilde{\boldsymbol{x}}-\boldsymbol{x}\|_2^2}{2\sigma^2})}{Z_{\boldsymbol{\theta}}(2\pi\sigma^2)^{\frac{n}{2}} p(\tilde{\boldsymbol{x}})} \\
&= \frac{\exp(-U_{\boldsymbol{\theta}}(\boldsymbol{x}) - \frac{\|\tilde{\boldsymbol{x}}-\boldsymbol{x}\|_2^2}{2\sigma^2})}{\tilde{Z}_{\boldsymbol{\theta}}(\tilde{\boldsymbol{x}})}
\end{aligned}$$

Suppose now that for our samples $(\boldsymbol{x}_i)_{i=1}^n$ we have the perturbed samples $(\tilde{\boldsymbol{x}}_i)_{i=1}^n$ that we gained by adding noise with $\tilde{\boldsymbol{x}}_i = \boldsymbol{x}_i + \sigma\varepsilon_i$, then the normalised recovery-log-likelihood (RLL) is defined as

$$\frac{1}{n}\mathcal{J}(\boldsymbol{\theta}) = \frac{1}{n}\sum_{i=1}^{n}\log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\tilde{\boldsymbol{x}})$$

With the expression above this is

$$\frac{1}{n}\sum_{i=1}^{n}\log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\tilde{\boldsymbol{x}}) = \frac{1}{n}\sum_{i=1}^{n}\left(-U_{\boldsymbol{\theta}}(\boldsymbol{x}) - \frac{\|\tilde{\boldsymbol{x}}-\boldsymbol{x}\|_2^2}{2\sigma^2}) - \log(\tilde{Z}_{\boldsymbol{\theta}}(\tilde{\boldsymbol{x}})\right)$$

where

$$\begin{aligned}\log(\tilde{Z}_{\boldsymbol{\theta}}(\tilde{\boldsymbol{x}})) &= \log(Z_{\boldsymbol{\theta}}(2\pi\sigma^2)^{\frac{n}{2}}p(\tilde{\boldsymbol{x}}))\\ &= \log(Z_{\boldsymbol{\theta}}) + \frac{n}{2}\log(2\pi\sigma^2) + \log(p(\tilde{\boldsymbol{x}}))\end{aligned}$$

Thus upon differentiating by θ we have

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\theta}}\frac{1}{n}\mathcal{J}(\boldsymbol{\theta}) &= -\frac{1}{n}\sum_{i=1}^{n}\frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\theta}}U_{\boldsymbol{\theta}}(\boldsymbol{x}) - \frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\theta}}\log(\tilde{Z}_{\boldsymbol{\theta}}(\tilde{\boldsymbol{x}})\\ &= -\frac{1}{n}\sum_{i=1}^{n}\frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\theta}}U_{\boldsymbol{\theta}}(\boldsymbol{x}) - \frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\theta}}(\log(Z_{\boldsymbol{\theta}}) + \frac{n}{2}\log(2\pi\sigma^2) + \log(p(\tilde{\boldsymbol{x}})))\\ &= -\frac{1}{n}\sum_{i=1}^{n}\frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\theta}}U_{\boldsymbol{\theta}}(\boldsymbol{x}) - \frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\theta}}\log(Z_{\boldsymbol{\theta}})\\ &= \frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\theta}}\frac{1}{n}\mathcal{L}(\boldsymbol{\theta})\end{aligned}$$

So the gradients of RLL and standard LL coincide. This means we can sample from the perturbed distribution $p_{\boldsymbol{\theta}}(\boldsymbol{x}|\tilde{\boldsymbol{x}})$, which is less multimodal and hence easier to sample from.
When we use the previously introduced Markov chain samplers we can treat the perturbed distribution like another EBM model with conditional energy functional

$$\tilde{U}_{\boldsymbol{\theta}}(\boldsymbol{x}) = U_{\boldsymbol{\theta}}(\boldsymbol{x}) + \frac{\|\tilde{\boldsymbol{x}}-\boldsymbol{x}\|_2^2}{2\sigma^2}$$

and use its gradient to generate the samples. This gradient is given by

$$\begin{aligned}\nabla_{\boldsymbol{x}}\tilde{U}_{\boldsymbol{\theta}}(\boldsymbol{x}) &= \nabla_{\boldsymbol{x}}(U_{\boldsymbol{\theta}}(\boldsymbol{x}) + \frac{\|\tilde{\boldsymbol{x}}-\boldsymbol{x}\|_2^2}{2\sigma^2})\\ &= \nabla_{\boldsymbol{x}}U_{\boldsymbol{\theta}}(\boldsymbol{x}) + \nabla_{\boldsymbol{x}}\frac{(\tilde{\boldsymbol{x}}-\boldsymbol{x})^T(\tilde{\boldsymbol{x}}-\boldsymbol{x})}{2\sigma^2}\\ &= \nabla_{\boldsymbol{x}}U_{\boldsymbol{\theta}}(\boldsymbol{x}) - \frac{\tilde{\boldsymbol{x}}-\boldsymbol{x}}{\sigma^2}\end{aligned}$$

9 Metrics

Recovery Likelihood promises an improvement in convergence speed by accelerating the calculation of the likelihood gradient. Does this improvement come at the cost of quality, reliability or increased maintenance? What would we like to have:

1. Accurate parameter estimates
2. A sufficiently close parametric representation/model for the data distribution
3. Consistent quality in training
4. Independence of the model
5. Straightforward parameter choice and low dependency for perturbation

Comparing the likelihood approaches and the involved samplers in a rigorous way requires some measure of both the quality of the final estimates and the reliability of the approach.

9.1 Metrics for Estimated Parameters

For comparing the quality of the estimates we can look for one at metrics of the distance between the known parameters and the estimated parameters. This gives a measure of how good the respective approach is at estimating the parameters accurately and would be specifically important in contexts, where the parameters have a specific meaning and the estimates are meant to be interpretable. For this vector norms from the p-norm family or matrix norms like the operator- or Frobenius-norm are suitable.