Ai-Sampler: *A*dversarial Learning of Markov kernels with *i*nvolutive maps

Evgenii Egorov [* 1]   Riccardo Valperga [* 2]   Efstratios Gavves [2]

Abstract

Markov chain Monte Carlo methods have become popular in statistics as versatile techniques to sample from complicated probability distributions. In this work, we propose a method to parameterize and train transition kernels of Markov chains to achieve efficient sampling and good mixing. This training procedure minimizes the total variation distance between the stationary distribution of the chain and the empirical distribution of the data. Our approach leverages involutive Metropolis-Hastings kernels constructed from reversible neural networks that ensure detailed balance by construction. We find that reversibility also implies C_2-equivariance of the discriminator function which can be used to restrict its function space.

1. Introduction

Markov Chain Monte Carlo (MCMC) is a key approach in statistics and machine learning when it comes to sampling from complex unnormalized distributions. MCMC generates samples by setting up a Markov chain that has the target distribution as its stationary distribution. Once converged, samples can be obtained by recording states from the chain. Not only have MCMC methods transformed Bayesian inference, allowing to sample from the untractable posterior distributions of complex probabilistic models, but they are also widely used for integral estimation, time-series analysis and many other problems in statistics and probabilistic modelling (Robert et al., 2004). Together with Variational Inference (Blei et al., 2017), they are the two primary probabilistic inference methods in machine learning and statistics. Compared to Variational Inference, MCMC methods are often less efficient because of their iterative nature, but more

applicable in general and, most importantly, they are asymptotically unbiased (Roberts & Rosenthal, 2004). In the recent years, the evolution of deep neural networks has notably propelled the field of Variational Inference (Rezende & Mohamed, 2015; Kingma & Welling, 2014; Rezende et al., 2014; Kingma et al., 2016) whilst MCMC methods have not benefited much from these advances. Using neural networks inside MCMC algorithms would not only result in more powerful and flexible inference methods, capable of handling complex, high-dimensional problems more effectively, but also exploit the tremendous advancements in efficiency of hardware specifically designed for neural network computations, such as GPUs and TPUs. We argue that this stagnation in developing MCMC algorithms that make use of neural networks is partly due to the well-known difficulty of measuring sample quality (see for example Gorham & Mackey (2015); Brooks et al. (2011)), meaning that systematically establishing the convergence of a chain to its target distribution, or defining a quality measure for samples from a Markov chain is challenging. This makes it intrinsically hard to define an objective function that can be optimized with stochastic gradient descent to improve the performance of the Markov chain. In general, the design of a suitable objective function poses a challenge as one must balance two competing goals: encouraging both *high-quality samples* and *good exploration* of the whole space (Levy et al., 2018). In light of this, we attempt to answer the simple question: *how can we learn to sample from a given unnormalized distribution, using neural network-based MCMC methods?* To answer, we propose a novel MCMC method that makes use of time-reversible neural networks for the transition kernel and derive an upper bound to the total variation distance between the stationary distribution of the resulting Markov chain and the target distribution. Our contribution can be summarized as follows: we first derive an MCMC method based on *reversible* neural networks that is trained as an implicit generative model using a novel adversarial objective from a given *empirical* data distribution. The proposed objective makes use of a discriminator. We prove the optimal discriminator to be equivariant with respect to the cyclic group of order 2 and propose a class of C_2-equivariant functions that can be used to parameterise it. Finally, we use a bootstrap process to learn to sample from a given analytic density and show, on various synthetic, and real-world examples, that the Markov chain resulting from the learned

---

[*]Equal contribution [1]Amsterdam Machine Learning Lab, University of Amsterdam, the Netherlands [2]Riccardo Valperga VIS-Lab, University of Amsterdam, the Netherlands. Correspondence to: Evgenii Egorov <egorov.evgenyy@ya.ru>, Riccardo Valperga <r.valperga@uva.nl>.

transition kernel produces samples from the target density much more effectively and efficiently than other existing methods.

2. Related Work

We highlight works that are close to ours in that they make use of neural networks to define MCMC methods. For general MCMC techniques, the reader is referred to comprehensive surveys such as Roberts & Rosenthal (2004); Brooks et al. (2011); Luengo et al. (2020). To address the challenge in the design of an objective for MCMC methods, previous works (Titsias & Dellaportas, 2019; Hirt et al., 2021; Roberts & Rosenthal, 2009) propose a specific type of deterministic proposal that targets a particular acceptance ratio while promoting mixing with entropy-like regularization. However, this approach imposes restrictions on the type of proposal used and introduces a hyperparameter, that is either the target acceptance rate or the weight of the regularization. Pasarica & Gelman (2010) develop an adaptive MCMC method that selects the parametric kernel that maximizes the expected squared jump. Another solution proposed by Levy et al. (2018) involves optimizing the difference between the average Euclidean distance after one step and its inverse. An interesting method that mixes Variational Inference with Hamiltonian Monte Carlo is the work of Hoffman et al. (2019) where autoregressive flows are used to correct unfavorable geometry of a posterior distribution. Samsonov et al. (2022) propose a method that uses both local and global samplers in which local steps are enclosed between global updates from an independent proposal. Similarly, (Gabrié et al., 2022) also propose an adaptive MCMC which augments sampling with non-local transition kernels parameterized with normalizing flows. A notable method for learning to sample, that is also the closest to our approach, is the work of Song et al. (2017) where a method for training Markov kernels parameterized using neural networks with an autoencoder architecture is proposed. It consists of a GAN-like objective justified by the assumption that GANs attempt to minimize the Jensen-Shannon divergence. Despite the empirical success, theoretical guarantees on GANs are hard to derive and most results assume optimality of the discriminator which in practice never holds (Goodfellow et al., 2020; Arjovsky et al., 2017). Furthermore, there is empirical evidence that GANs do not actually sample from a target distribution (Arora & Zhang, 2017). Similar to our approach, their method involves a bootstrap process where the quality of the Markov chain kernel increases over time. One key difference is that, to achieve reversibility, Song et al. (2017) make use of an additional random variable, whereas our parameterized deterministic proposals are reversible *by construction*.

3. Parametrizing Kernels with Involutions

MCMC algorithms are defined by a transition kernel $t(x'|x)$ that maps probability density functions [1] $p_t(x)$ to other probability density functions: $p_{t+1}(x') = \int_{\mathcal{X}} t(x'|x)p_t(x)dx$. The probability density $p(x)$ is *stationary* for the Markov kernel $t(x'|x)$ if

$$\int_{\mathcal{X}} t(x'|x)p(x)dx = p(x'). \tag{1}$$

Reversible kernels, namely kernels for which the *detailed balance* condition holds:

$$t(x'|x)p(x) = t(x|x')p(x'), \forall x, x' \in \mathcal{X} \times \mathcal{X}, \tag{2}$$

have p as stationary probability distribution. Metropolis-Hastings kernels is a popular class of such kernels based on a proposal mechanism and a density ratio acceptance function. Throughout the rest of the paper we consider the following transition kernel that satisfies detailed balance with respect to a given probability distribution p:

Definition 3.1. (Neklyudov et al., 2020) Given a distribution $p(x), x \in \mathcal{X}$ and a deterministic map $M : \mathcal{X} \to \mathcal{X}$, such that $M \circ M = id_{\mathcal{X}}$, the *involutive Metropolis-Hastings kernel* is

$$t(x'|x) := \delta(x' - Mx) \min\left(1, \frac{p(Mx)}{p(x)} J_x^M\right) +$$
$$+ \delta(x' - x)\left(1 - \min\left(1, \frac{p(Mx)}{p(x)} J_x^M\right)\right), \tag{3}$$

where J_x^M is the absolute value of the determinant of the Jacobian of M at point x.

As specified in the definition, for this transition kernel to satisfy the *fixed point equation* (1) the deterministic map M must be *involutive*. Given that the kernel is deterministic, this condition would obviously restrict our chain to transition between x and $x' = Mx$. In order to cover the whole support of $p(x)$ we introduce *auxiliary variables* $v \in V$. Then, instead of sampling from $p(x)$ we sample from $p(x, v) = p(x)p(v|x)$, with $p(v|x)$ being any probability distribution we can efficiently sample from.

It can be shown (Neklyudov et al., 2020) that the transition kernel of the Markov chain from such algorithm satisfies detailed balance (2) with respect to $p(x, v)$ and fixed-point equation (1) with respect to $p(x)$.

The involutive MCMC framework formulates MCMC algorithms in terms of two degrees of freedom: the involution $M : \mathcal{X} \times V \longrightarrow \mathcal{X} \times V$, and the conditional distribution

---

[1]Throughout the paper we use the words *density* and *distribution* interchangeably, since we assume every distribution is abs. continuous w.r.t. the Lebesgue measure.

$p(v|x)$. Since we want to train a sampler that samples optimally, we can define a family of parameterized involutions M_θ for $\theta \in \Theta$, and we would like to optimize for θ to obtain a Markov chain that efficiently and effectively samples from a target density. In determining what is efficient and effective sampling, defining what is a natural objective is harder than it sounds. On one hand, the acceptance ratio of new proposals quantifies efficiency, on the other hand, the *mixing of the chain* that quantifies effectiveness is just as important yet hard to define. Finding a good compromise between the two sub-objectives is difficult. To address this, we design an adversarial game between the mapping M_θ and a *discriminator*.

4. Parameterizing Involutions

Before delving into the details of training an involutive transition kernel, in this section, we discuss the parameterization of the deterministic involution M_θ. As pointed out in Song et al. (2017) it is difficult to parameterize deterministic involutions: if the proposal is deterministic, then for all $(x, v) \in \mathcal{X} \times V$ we should have $M_\theta(M_\theta(x, v)) = (x, v)$, which is not straightforward to impose non trivially as a constraint in the parameterization. As a solution they introduce a third auxiliary random variable u sampled from a uniform distribution $U(0, 1)$ and either use the forward or the inverse of M_θ, depending on the value of u, to obtain the proposal. Recognizing that time-reversibility of deterministic dynamical systems and detailed balance of Markov chains are related, we propose as an alternative to use a class of neural networks known as *time-reversible neural networks*.

4.1. Time-reversible neural networks

In the context of physics-informed learning and forecasting of Hamiltonian systems, Valperga et al. (2022) introduced time-reversible neural networks. These architectures are good candidates for parameterizing involutions. To motivate this, we first review how (time) reversing symmetries relate to Markov Chains and detailed balance, especially in the context of flow maps in Hamiltonian Monte Carlo. We then describe how we can further decompose uniquely the time-reversible map and parameterize it with a universal approximators like neural networks.

Reversing symmetries in Hamiltonian MC kernels. We say that an invertible smooth map $R : \mathcal{X} \times V \to \mathcal{X} \times V$ is a reversing symmetry for an invertible function $L : \mathcal{X} \times V \to \mathcal{X} \times V$ if

$$R \circ L \circ R = L^{-1}. \tag{4}$$

In particular, we call *time*-reversing symmetry the linear map

$$R : (x, v) \mapsto (x, -v). \tag{5}$$

If R is a reversing symmetry of a function we call the function R-*reversible*. Time-reversing symmetry is a typical property of autonomous Hamiltonian systems. It is particularly important when constructing Markov chain samplers out of Hamiltonian systems in Hybrid Monte Carlo algorithms (HMC), where the function L is the deterministic flow of a Hamiltonian system, defined using the target density (Duane et al., 1987; Neal, 2011). In HMC the deterministic proposal is obtained by negating the momenta after applying the deterministic flow. From Eq. 4, it follows that the resulting map $R \circ L$ is indeed an involution, since $(R \circ L) \circ (R \circ L) = (L^{-1} \circ R^{-1}) \circ (R \circ L) = \mathrm{id}_{\mathcal{X} \times V}$, which ensures that detailed balance is satisfied with respect to the target density.

Decomposing and parameterizing reversibile maps. Valperga et al. (2022) provide a method for defining parametric functions that, by construction, are reversible with respect to any linear reversing symmetry. In particular, the following theorem holds

Theorem 4.1. *(Valperga et al., 2022) Let $L : \mathbb{R}^D \to \mathbb{R}^D$ be an R-reversible diffeomorphism[2], with R being a linear involution. Then, there exists a unique diffeomorphism $g : \mathbb{R}^D \to \mathbb{R}^D$, such that $L = R \circ g^{-1} \circ R \circ g$. If L is symplectic, then g can be chosen symplectic.*

This theorem ensures that any R-reversible, or in general R-reversible and symplectic map L, can be decomposed as $L = R \circ g^{-1} \circ R \circ g$. This shifts the problem from that of approximating L to that of approximating the *unique g* of its decomposition. At this point, g can be approximated using any universal approximator with the only constraint that we need the *analytic* form of the inverse g^{-1}. Suitable candidates are, for example, compositions of real NVP bijective layers (Dinh et al., 2017), or, as done in Valperga et al. (2022), compositions of Hénon maps (see Appendix A).

Involutive MCMC kernels by construction. Using the above result together with definition 3.1 for the involutive Metropolis-Hastings kernels, we can construct the parametric family of deterministic proposals $M_\theta : \mathcal{X} \times V \to \mathcal{X} \times V$

$$M_\theta = R \circ L_\theta, \text{ with } L_\theta = R \circ g_\theta^{-1} \circ R \circ g_\theta, \tag{6}$$

where R is is the time-reversing symmetry of Eq. (5), so that $M_\theta(M_\theta(x, v)) = (x, v)$ for all $(x, v) \in \mathcal{X} \times V$.

5. Adversarial Training for Involutive Kernel

Having defined a parameterization for an involutive map M_θ, we are now interested in how to train M_θ using the

---

[2]It must be smoothly isotopic to the identity, a mild condition for sufficiently well-behaved target functions.
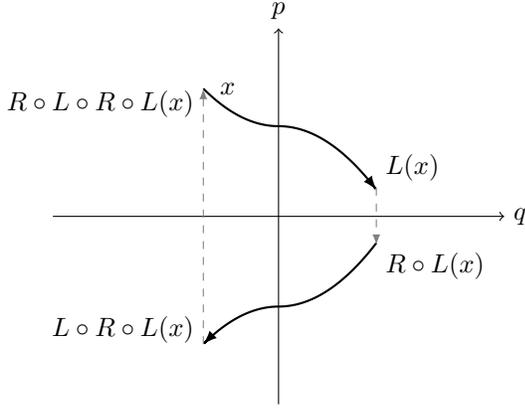
Figure 1. Schematic representation of an involution constructed from a time-reversible diffeomorphism L. For an R-reversible diffeomorphism L, with $R : (q, p) \mapsto (q, -p)$, the composition $R \circ L \circ R \circ L$ is the identity.

transition kernel in equation (3). The core idea is to train the parametric transition kernel as a *generative model* that samples from the target distribution. Therefore, the objective that we derive in this section allows for unbiased estimation with samples from the target distribution. To retrieve samples from the target distribution we make use of a *bootstrap* process.

5.1. Bootstrap

An unbiased estimator of the objective derived in this section can be computed as a Monte Carlo sum over samples drawn *from the target distribution*. Similarly to Song et al. (2017), to get samples from the target distribution and train our transition kernel we propose a bootstrap process that gradually increases the quality of samples over time. We first obtain samples from $p(x)$ using a possibly inefficient and slow-mixing kernel that nonetheless has $p(x)$ as its stationary distribution. We then use these samples to train our kernel and get new samples of higher quality. By repeating this process the quality of samples increases, which in turns leads to a better transition kernel. The bootstrap process is outlined in Algorithm 1.

5.2. The adversarsial MH kernel

Let us assume that we are given samples from the target distribution obtained with the bootstrap process as just described. Given samples and the target density ratio, simply training M_θ to maximize the expected acceptance rate would lead to poor exploration. For example, $M_\theta = id_{\mathcal{X} \times V}$ would be a trivially optimal solution for which the sampler maximizes the acceptance rate by simply remaining in the same location. As done in Levy et al. (2018) we could also minimize the lag-one autocorrelation, which is equivalent to

maximizing expected squared transition distance. However, balancing between expected acceptance rate and expected distance is hard.

To train the transition kernel M_θ, rather than using the target distribution directly to accept or reject a proposed sample, we delegate the acceptance and rejection of proposed samples to a learnable discriminator network that is trained to optimally accept samples. In place of the true density ratio $\frac{p(x')}{p(x)}$ we introduce a discriminator $D(x)$ as an approximation. We then define an adversarial objective between two terms. The first term is about training the parametric proposal $R \circ L_\theta$ that proposes novel x'. The second term is about optimizing the parametric discriminator D to correctly tell apart the proposed novel samples from those that have been drawn from $p(x)$.

The overall objective is adversarial. On one hand, we want to optimize the proposal mapping to return novel proposals that fool the discriminator to believe they are samples from the true density. On the other hand, we want to optimize the discriminator to not be fooled and recognize novel proposals. By optimizing this adversarial objective successfully, we obtain a Markov transition kernel that effectively samples from the target distribution.

From now on we identify $\mathcal{S} = \mathcal{X} \times V$ with $\mathbb{R}^{2n}$ and consider the auxiliary noise to be included in $x \in \mathcal{S}$.

Definition 5.1. Let $R \circ L_\theta : \mathcal{S} \to \mathcal{S}$ be a deterministic involutive map, and $D : \mathcal{S} \to \mathbb{R}_+$ be a positive valued deterministic function. For a generic test function $r : \mathbb{R}_+ \to [0, 1]$, such that $x \cdot r(\frac{1}{x}) = r(x)$, and $r'(x) \geq 0$, we define the *Adversarial Metropolis-Hastings transition kernel* as:

$$t_D(x'|x) = \delta(x' \text{-} RL_\theta(x))r\left[D(x)\right] + \delta(x' \text{-} x)(1 \text{-} r\left[D(x)\right]).$$

$$(7)$$

An example of a test function is $r(x) = \min(1, x)$. We use the Barker test: $r_{\text{B}}(x) = \left(1 + \frac{1}{x}\right)^{-1}$ during training proposal and common $\min(1, x)$ during sampling from trained proposal. In order to sample from the target distribution, we need to satisfy detailed balance, $t_D(x'|x)p(x) = t_D(x|x')p(x')$, with respect to the target distribution. We propose to ensure this by minimising the distance between the target distribution $p(x)$ and the distribution obtained after one step of the chain starting from the target, that is $t_D \circ p(x)$. As done in Neklyudov et al. (2019), we consider the *total variation* (TV) distance:

$$TV[p, t_D \circ p(x)] := \frac{1}{2} \int_{\mathcal{S}} |p(x) - t_D \circ p(x)| dx. \quad (8)$$

For any kernel t_D as in Def. 5.1 we have $TV\left[p(x'), t_D \circ p(x)\right] = 0$ if $\log D(x) = \log \frac{p(R \circ L_\theta(x))}{p(x)} J_x^{R \circ L_\theta}$. Given that $R \circ L_\theta$ is an involution, we can show that the optimal log-discriminator

function has a simple symmetry under the action of $R \circ L_\theta$. We describe next the symmetry and how to include it to derive our final objective.

5.3. Equivariance of the discriminator under $R \circ L_\theta$

Let[3] $R \circ L_\theta$ be an involutive map with R volume and density-preserving, i.e., such that $p(RL_\theta(x)) = p(x)$. It's immediate to verify that the density ratio $\lambda(x) = \frac{p(RL_\theta(x))}{p(x)} J_x^{RL_\theta}$ has the following symmetry:

$$\lambda(RL_\theta(x)) = \frac{p(RL_\theta \circ RL_\theta(x))}{p(I \circ RL_\theta(x))} J_{RL_\theta(x)}^{RL_\theta} =$$
$$= \frac{p(x)}{p(RL_\theta(x))} \left(J_x^{RL_\theta}\right)^{-1} = (\lambda(x))^{-1}, \tag{9}$$
$$\log \lambda(RL_\theta(x)) = -\log \lambda(x).$$

where we used $1 = J_{RL_\theta(x)}^{RL_\theta} J_x^{RL_\theta}$, and $(RL_\theta)^{-1} = RL_\theta$. Therefore, under the action of RL_θ, we would like the parametric discriminators D to transform, *by construction*, similar to the density ratio λ.

We enforce the above constraint using C_2-equivariant functions. Let us consider the *cyclic group* $C_2 = \langle g, g^2 = e \rangle$. Note that the action of the *generally non-linear* involution RL_θ is *linear* if we consider its action on the lifted space: $x \oplus \left(R \circ L_\theta(x)\right) \cong \mathbb{R}^{2n} \oplus \mathbb{R}^{2n}$. In this space, RL_θ is the representation ρ_{2n} of the C_2 group:

$$\rho_{2n} : C_2 \to GL(\mathbb{R}^{2n} \oplus \mathbb{R}^{2n}), \ \rho_{2n}(g) = \begin{bmatrix} 0 & I_{2n} \\ I_{2n} & 0 \end{bmatrix}. \tag{10}$$

We also consider another representation of C_2 given by sign flip:

$$\xi_1 : C_2 \to GL(\mathbb{R}), \ \xi_1(g) = -1. \tag{11}$$

Next we describe parameterizations for the discriminator using C_2-equivariant functions.

5.4. Discriminator parametrization

To enforce the desired transformation property we parameterize the logarithm of the discriminator with a neural network $d_\phi(x) = \log D_\phi(x)$. The goal is to approximate the log-density ratio: $d_\phi(x) \approx \lambda(x)$, subject to equivariance constraints. We propose two possible parameterizations: a simple product parameterization, and a more general composition of linear maps and non-linear activations.

Discriminator with product parameterization. A special construction of C_2-equivariant discriminator is with functions $f : \mathbb{R}^{2n} \oplus \mathbb{R}^{2n} \to \mathbb{R}$ that are equivariant under ρ_{2n} and ξ_1, namely such that

$$f(\rho_{2n}x) = \xi_1 f(x). \tag{12}$$

Any equivariant function $d : \mathbb{R}^{2n} \oplus \mathbb{R}^{2n} \to \mathbb{R}$ can be decomposed into an equivariant and an invariant part [4]. We can then write the equivariant part as the difference of any function $\eta : \mathbb{R}^{2n} \to \mathbb{R}$ computed at x and $RL_\theta(x)$. The invariant part can be any function $\psi : \mathbb{R}^{2n} \to \mathbb{R}$ of the sum $x + RL_\theta(x)$:

$$d_\phi(x) = \psi(x + RL_\theta(x))[\eta(RL_\theta(x)) - \eta(x)]. \tag{13}$$

Equivariance follows from symmetry of function construction:

$$d_\phi(RL_\theta(x)) = \psi(RL_\theta(x) + x)[\eta(x) - \eta(RL_\theta(x)] =$$
$$= -d_\phi(x). \tag{14}$$

This constructions is a special case of composition lifting layer with specific fixed linear layer. Below we consider general case.

C_2-**equivariant composition of linear maps and non-linear activatons.** The more general construction for a C_2-equivariant discriminator is with functions $h : \mathbb{R}^{2n} \oplus \mathbb{R}^{2n} \to \mathbb{R} \oplus \mathbb{R}$, which are equivariant under ρ_{2n} and ρ_1:

$$h(\rho_{2n}x) = \rho_1 h(x). \tag{15}$$

Let $\hat{d}_\phi$ be a two-channel neural network that we use to approximate the logarithm of the density ratio at both the pre-image and image of RL_θ:

$$\hat{d}_\phi(x) \approx \begin{bmatrix} \log \frac{p(RL_\theta(x))}{p(x)} J_x^{RL_\theta} \\ -\log \frac{p(RL_\theta(x))}{p(x)} J_x^{RL_\theta} \end{bmatrix}. \tag{16}$$

Then, let us consider a linear map $\begin{bmatrix} A & B \\ C & D \end{bmatrix} : \mathbb{R}^{2n} \oplus \mathbb{R}^{2n} \to \mathbb{R}^{2s} \oplus \mathbb{R}^{2s}$, for some $s \leq n$. To obtain functions that are equivariant with respect to ρ_{2n} and ρ_{2s} from compositions of such linear maps they must satisfy the following constraint for all x:

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix} \begin{bmatrix} 0 & I_{2n} \\ I_{2n} & 0 \end{bmatrix} \begin{bmatrix} RL(x) \\ x \end{bmatrix} = \begin{bmatrix} 0 & I_{2s} \\ I_{2s} & 0 \end{bmatrix} \begin{bmatrix} A & B \\ C & D \end{bmatrix} \begin{bmatrix} RL(x) \\ x \end{bmatrix}. \tag{17}$$

which is equivalent to setting $A = D$, $B = C$. We can then compose linear layers of this form with element-wise non-linearities. The function $d_\phi(x)$, that we use to approximate the log-density ratio can then just be the first, or the second, coordinate of the two-dimensional output of $\hat{d}_\phi$.

At this point it is important to notice that in both the proposed parameterizations, the discriminator D depends not only on the parameters ϕ, but also on θ through the proposal map RL_θ.

---

[3]from now on, where needed for compactness, we omit the composition sign '$\circ$' and simply use juxtaposition.

[4]This is trivial since any scalar equivariant function remains equivariant if multiplied by an invariant function.

5.5. Detailed-Balance Loss

In order to simplify equations, and for readability, we will consider the case of volume-preserving maps RL_θ. For the general case, the derivations are the same, but with the difference that the Jacobians of RL_θ are absorbed by the density ratio.

As anticipated in the previous sections, we need to minimise $TV[p, t_D \circ p]$. Let t_D be the transition kernel of the type (5.1) with discriminator $D = \exp(d)$. Given the target density $p(x)$ the total variation distance between p and $t_D \circ p$ is[5]

$$TV[p; t_D \circ p] = |p(RL_\theta(x))r[D(RL_\theta(x))] - p(x)r[D(x)]|_1 \tag{18}$$

We consider an upper bound on this quantity which is suitable for optimisation and has the same optimum points.

Upper Bound on TV with Pinsker Inequality We use the famous Pinsker Inequality (Pinsker, 1964) to upper-bound the TV distance with a more malleable KL divergence. Given two densities p and q we have

$$TV[p; q]^2 \le \mathrm{KL}[p; q]. \tag{19}$$

Since Pinsker Inequality is defined for probability densities, we take an intermediate step and normalize the functions:

$$TV[p; t_D \circ p] = A \cdot TV[p^\infty; p^\infty_{R \circ L_\theta}], \tag{20}$$

where

$$\begin{aligned} A &= \int_{\mathcal{S}} p(x)r[D(x)]\,dx, \, A \in [0; 1], \\ p^\infty &= A^{-1}p(x)r[D(x)], \\ p^\infty_{RL_\theta} &= A^{-1}p(RL(x))r[D(RL(x))]. \end{aligned} \tag{21}$$

These equations link the detailed balance condition to rejection sampling since p^∞ is the density induced by rejection sampling with proposal distribution $p(x)$ and acceptance function $r[D(x)]$. Now we can apply Pinsker inequality:

$$\begin{aligned} &TV^2[p^\infty; p^\infty_{RL}] \le A^2 \cdot \mathrm{KL}[p^\infty; p^\infty_{RL}], \\ &A^2 \mathrm{KL}[p^\infty; p^\infty_{RL}] = \\ &A \cdot \mathbb{E}_{p(x)}\left(r[D(x)]\left(\log \frac{p(x)}{p(RL(x))} + \log r[D(x)] \right) \right), \\ &TV^2[p; t_D \circ p] \le \\ &A \cdot \mathbb{E}_{p(x)}\left(r[D(x)]\left(\log \frac{p(x)}{p(RL(x))} + \log r[D(x)] \right) \right) \le \\ &A \cdot \left(\mathrm{KL}[p(x); p(RL(x))] + \mathbb{E}_{p(x)}\left(r[D(x)] \log r[D(x)] \right) \right). \end{aligned} \tag{22}$$

---

[5]We use $|\cdot|_1$ for $\int_{\mathcal{S}} |\cdot| dx$

Considering the final bound, we have the following *adversarial* optimization problem over the parameters θ of RL_θ and the parameters ϕ of D_{ϕ, RL_θ}:

$$\max_\theta A_\theta = \max_\theta \mathbb{E}_{p(x)}\left(r[D_{\phi, RL_\theta}(x)] \right), \text{with fixed } \phi$$

$$\min_\phi \mathbb{E}_{p(x)}\left(r[D_{\phi, RL_\theta}(x)] \log r[D_{\phi, RL_\theta}(x)] \right), \text{with fixed } \theta. \tag{23}$$

As already mentioned, due to the equivariance constraints, $D(x)$ is a function of both θ, through RL_θ, and ϕ. For any fixed RL_θ the solution of the minimisation problem is the log-density ratio:

$$D^*(x) =$$
$$= \arg\min \int p(x)r[D_{\phi, RL_\theta}(x)] \log r[D_{\phi, RL_\theta}(x)]\,dx$$
$$= \log \frac{p(RL_\theta)}{p(x)} J_x^{RL_\theta(x)}. \tag{24}$$

For such a discriminator $TV^2[p; t_{D^*} \circ p] = 0$ since detailed balance with respect to p is satisfied. However, in the beginning the discriminator is not optimal. Instead, we start optimizing from a randomly initialised discriminator and iteratively improve D_ϕ and $R \circ L_\theta$ in alternating steps: optimising over θ makes the problem harder for the discriminator and vice versa (see Section 6.1).

It is worth noting from Eq. (22) that the tightness of our bound depends on $\mathrm{KL}[p(x); p(R \circ L_\theta(x))]$. However, if $R \circ L_\theta$ is close to being density preserving, this term vanishes.

5.6. Optimization algorithms

The objectives (23) are defined as expectations over the given empirical distribution $p(x) = \frac{1}{N} \sum_{n=1}^{N} \delta(x - x_n)$. Therefore, they allow for unbiased gradient estimates:

$$\nabla_\theta \mathbb{E}_{p(x)}\left(r[D_{\phi, R \circ L_\theta}(x)] \right) = \mathbb{E}_{p(x)} \nabla_\theta \left(r[D_{\phi, R \circ L_\theta}(x)] \right)$$

$$\approx \frac{1}{B} \sum_{b=1}^{B} \nabla_\theta \left(r[D_{\phi, R \circ L_\theta}(x_b)] \right), \, \{x_b\}_{b=1}^{B}, \, x_b \sim p(x), \tag{25}$$

and similarly for the gradient with respect to ϕ. With the unbiased gradient estimates, we use standard Adam optimizer with default hyperparameters and constant learning rate. With adversarial optimization it is often important to appropriately choose the learning schedule for the number of gradient descent steps taken for ϕ and θ. We found that the discriminator requires more gradient descent steps, e.g., a 1:3 or 1:2 ratio.

6. Experiments

We provide code for reproducing the experiments at https://github.com/ricvalp/aisampler. Following

Algorithm 1 Ai-sampler.

Input: target $p(x)$, initial kernel and discriminator T_θ, d_ϕ, initial $\mathcal{X} = \{x_j\}_{j=1}^N$, $x_j \sim p_0(x)$, $DiscSteps$, $KernelSteps$.
repeat
    $\mathcal{X} = \mathrm{MH}(T_\theta, p, \mathcal{X}, N)$ # MH is the Metropolis-Hastings algorithm.
    for X $\in \mathcal{X}$ **do**
        for $i = 1$ **to** $KernelSteps$ **do**
            $\theta \to \theta + \epsilon \nabla \mathcal{L}_A(X, \theta, \phi)$ # Eq. (23), first line
        end for
        for $i = 1$ **to** $DiscSteps$ **do**
            $\phi \to \phi + \epsilon \nabla \mathcal{L}_{adv}(X, \theta, \phi)$ # Eq. (23), second line
        end for
    end for
until convergence
Return: T_θ

Figure 2. Synthetic 2D densities used in the experiments. From left to right: *mog2*, *mog6*, *ring*, and *ring5*. Top row: true density. Bottom row: KDE with samples from our Ai-sampler.

Song et al. (2017), we test our method with the following experiments. The first experiment is with four 2D densities. They highlight specific challenges, such as good mixing in the presence of multiple modes separated by low density regions. The second experiment is with high-dimensional complex densities from real-world scenarios. In particular, we sample from the posterior of a logistic regression model trained with three different datasets with varying number of covariates. For both experiments we benchmark running time and efficiency of sampling of our method.

Baselines. For all the experiments we compare with Hamiltonian Monte Carlo (HMC) (Duane et al., 1987; Neal, 2011) and the method by Song et al. (2017), given its conceptual similarities with ours. For HMC we follow Song et al. (2017) and fix the number of leapfrog integration steps to 40 and tune the step-size to achieve the best performance. A more detailed description of the experiments, including the analytic expression of the densities can be found in the Appendix A and B.

Evaluation criteria. To compare the performances we use the *effective sample size* (ESS). Practically, the ESS is an estimate of the number of samples required to achieve the same level of precision that a set of uncorrelated random samples would achieve (for details see Appendix C). We report the lowest ESS among all covariates averaged over several trials. Since MCMC methods can be very costly to run, another often reported performance measure is the *effective sample size per second* (ESS/s). This is simply the ESS obtained per unit of time. We report this measure to evaluate efficiency. Despite being a common performance metric, a chain can achieve good ESS without generating good samples from the posterior. To make sure that this is
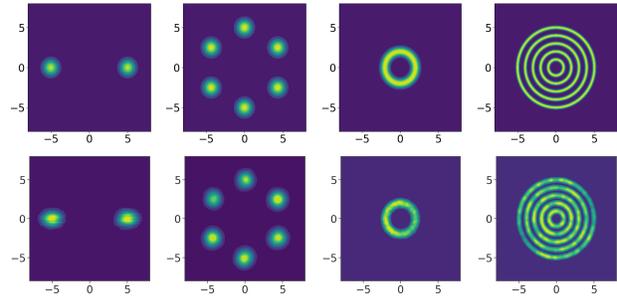
not the case, for the Bayesian logistic regression experiment we split each dataset into train and test subsets and compute the average log posterior using samples obtained with the train subset (see Appendix B). Furthermore, we compare the mean estimates with the ground truth ones.

6.1. 2-dimensional densities

For a fair comparison with Song et al. (2017) we choose to use the same four 2D densities they used. Two mixtures of Gaussians with two and six modes, *mog2* and *mog6*, a ring-shaped distribution, *ring*, and one made of five concentric rings, *ring5*. The densities are depicted in Fig. 2. We ran a single chain for 1000 burn-in steps and compute ESS and ESS/s for the following 1000 steps. Despite being 2D, these densities pose some challenges. In particular, the mixtures of Gaussians, and the concentric rings are multimodal distributions with high-density regions separated by high-energy (low-density) barriers. This characteristic represents a significant hurdle for Hamiltonian Monte Carlo, as Hamiltonian dynamics are unlikely to overcome these high-energy barriers, potentially leading to inefficient exploration of the state space and convergence issues. Figure 3 shows the very different behaviour of our method compared to HMC highlighting the fast mixing of our method compared in the presence of high energy barriers. Results are reported in Table 1.

Unbalanced modes. As as an additional qualitative experiments we follow (Samsonov et al., 2022) and investigate how our method performs on multimodal distributions with unbalanced modes. In particular we use a 3-component mixture of 2-dimensional isotropic Gaussians, with modes centered at the vertices of an equilateral triangle. We find that our model can effectively sample from unbalanced mixtures and give quantitatively estimate the weights of the modes. See Fig 8 in the Appendix.
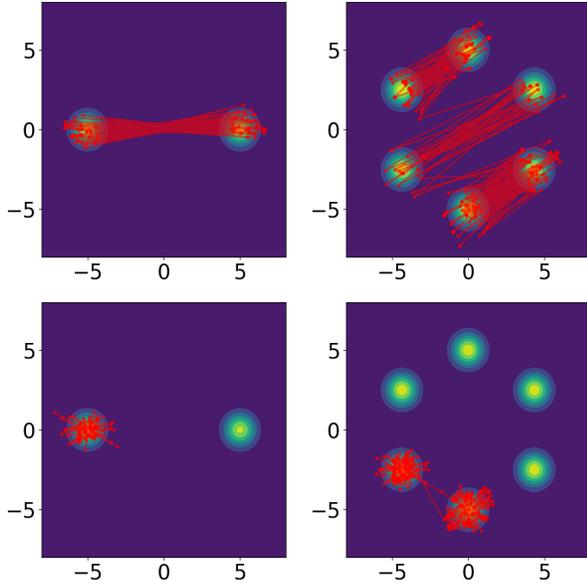
Figure 3. Single MCMC trajectory with the learned kernel (top) on the *mog2* and *mog6* synthetic 2D densities, compared to HMC (bottom). The low density regions make it unlikely for HMC to get from one mode to another.

Table 1. Effective sample size for synthetic 2D energy functions.

Density	ESS		
	HMC	A-NICE-MC	Ai-sampler (ours)
mog2	0.8	355.4	**1000.0**
mog6	2.4	320.0	**1000.0**
ring	981.3	**1000.0**	378.0
ring5	256.6	155.57	**396.5**

The role of the discriminator. We can investigate how the discriminator is "guiding" the parametric proposal by looking the value of $d_\phi(x)$ after training. In particular, we artificially turn d into a function of two points. For example, for the product parameterization we look at

$$d_\phi(x, y) = \psi(x + y)[\eta(y) - \eta(x)], \quad (26)$$

for a fixed x and different values of y. Figure 4 shows a discriminator, trained with *mog6* for three different values of x. Note that, for the discriminator to be effective it only needs to be consistent with the ground truth density ratio where the deterministic proposal is likely to propose the new sample from the current state of the chain.

6.2. Multi-dimensional densities

Bayesian logistic regression To compare with Song et al. (2017) we use the same posterior distribution they used, resulting from a Bayesian probabilistic logistic regression model on three famous datasets: *heart* (14 covariates, 532
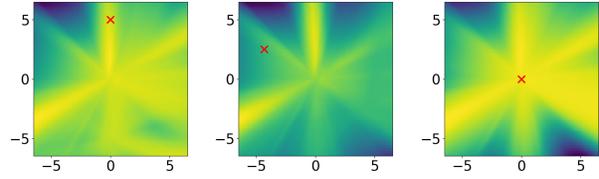

Figure 4. Discriminator as a function of two inputs as in Eq. (26), for three different values of x: one far from the six modes and two at the center of one mode.

Table 2. Averaged predictive posterior and l_2 norm of the difference between ground truth mean posterior and estimates from the chain.

Dataset	Log-Predictive posterior		Mean matching	
	HMC	Ours	HMC	Ours
Heart	-0.479	-0.458	5.4×10^{-5}	1.2×10^{-4}
German	-0.484	-0.484	1.3×10^{-5}	8.2×10^{-6}
Australian	-0.375	-0.375	1.2×10^{-5}	5.6×10^{-5}

data points), *australian* (15 covariates, 690 data points), and *german* (25 covariates, 1000 data points). For all experiments we ran a single chain for 1000 burn-in steps and compute ESS and ESS per second. for the following 5000 steps. Table 3 reports the results and we report average log posterior at Table 2.

Table 3. Effective sample size for Bayesian logistic regression.

Dataset	ESS		
	HMC	A-NICE-MC	Ai-sampler (ours)
Heart	**5000.0**	1251.2	**5000.0**
German	**5000.0**	926.49	**5000.0**
Australian	1113.4	1015.8	**1746.5**

6.3. Benchmarking running time

The advantage of learning to sample with a transition kernel parameterized by a neural network also lies in the hardware being optimized to perform multiple operations in parallel. Our method consists of a deterministic proposal parameterized by a neural network and, as opposed to HMC, does not require the gradient of the target density function. In HMC, proposals are obtained by integrating Hamiltonian dynamics which requires multiple calls to the gradient function of the density. For complex distributions, such as Bayesian logistic regression posteriors with large datasets, calls to the gradient are costly. We implement both our Ai-sampler and HMC in JAX (Bradbury et al., 2018) and Flax (Heek et al., 2023), making use of the built-in autodifferentiation tools,

vectorization, and just-in-time (JIT) compilation. We note that, in a certain range, running time remains approximately constant as the number of parallel chains increases. Past that range, running time increases approximately linearly. We measure running time within the constant range. For both HMC and the Ai-sampler we run Gelman's $\hat{R}$ diagnostic (Brooks & Gelman, 1998) and find values that suggest good convergence (≤ 1.004), and close-to-zero cross-correlation between chains. We can then assume that the ESS of parallel chains is the sum of the ESS of the single chains. For this reason, for a better comparison, we decided to report the ESS per second *per chain*. It is worth noting that, as reported in Fig. 5, we found that the Ai-sampler, given the relatively simple architecture, can sustain many more parallel chains than HMC, which would result in much larger overall ESS. For further details see the Appendix D.

Figure 5. Time vs. number of parallel chains for a single RTX3090 GPU, sampling from the Bayesian logistic regression posterior with German dataset. Every chain consists of 100 steps. For more more than 10^4 parallel chains, the *jitting* time becomes prohibitively long and therefore not worthy.

Table 4. Effective sample size per second per chain for the 2-dimensional densities.

Density	ESS/s	
	HMC	Ai-sampler (ours)
mog2	0.4	**1052.6**
mog6	0.98	**1041.7**
ring	**2725.8**	402.1
ring5	333.2	**434.7**

Table 5. Effective sample size per second per chain for Bayesian logistic regression.

Density	ESS/s	
	HMC	Ai-sampler (ours)
Heart	989.0	**1736.0**
German	672.0	**1618.0**
Australian	171.95	**1724.0**

7. Conclusions

We propose the *Ai-sampler*: an MCMC method with involutive Metropolis-Hastings kernels parameterized by time-reversible neural networks to ensure detailed balance. We derive equivariance conditions for the discriminator and a novel simple objective to train the parameterized kernel. The proposed objective is an upper-bound on the total variation distance between the target distribution and the stationary distribution of the Markov chain. We use the C_2-equivariance of the optimal discriminator to restrict the hypothesisis space of the parametric discriminators. We *learn to sample* with a bootstrap process that alternates between generating samples from the target density and improving the quality of the kernel with the adversarial objective. We demonstrate good mixing properties of the resulting Markov chain on some synthetic distributions and
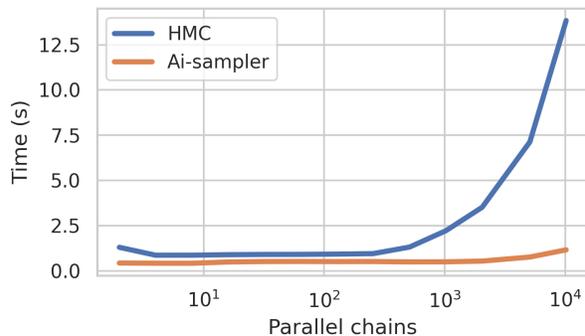
Bayesian inference with real-wold datasets. In the future, we plan to further explore the potential of our approach as a generative model, given that the adversarial objective is computed from an *empirical distribution*. In this regard, we would like to point out the similarity of our work with Normalizing Flows. In the generative model setting NFs have shown good results and our approach can be applied there two. We expect promising results, as instead of learning a global transformation from a simple probability distribution to a more complex one, e.g., transforming Gaussian noise into good-looking images, our approach consists of learning local transition kernels that map points (e.g., images) to other points in the target distribution.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

References

Arjovsky, M., Chintala, S., and Bottou, L. Wasserstein generative adversarial networks. In Precup, D. and Teh, Y. W. (eds.), *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pp. 214–223. PMLR, 2017. URL http://proceedings.mlr.press/v70/arjovsky17a.html.

Arora, S. and Zhang, Y. Do gans actually learn the distribution? an empirical study. *arXiv preprint arXiv:1706.08224*, 2017.

Blei, D. M., Kucukelbir, A., and McAuliffe, J. D. Variational inference: A review for statisticians. *Journal of the American statistical Association*, 112(518):859–877, 2017.

Bradbury, J., Frostig, R., Hawkins, P., Johnson, M. J., Leary, C., Maclaurin, D., Necula, G., Paszke, A., VanderPlas, J., Wanderman-Milne, S., and Zhang, Q. JAX: composable transformations of Python+NumPy programs. 2018. URL http://github.com/google/jax.

Brooks, S., Gelman, A., Jones, G., and Meng, X.-L. *Handbook of markov chain monte carlo*. CRC press, 2011.

Brooks, S. P. and Gelman, A. General methods for monitoring convergence of iterative simulations. *Journal of computational and graphical statistics*, 7(4):434–455, 1998.

Dinh, L., Sohl-Dickstein, J., and Bengio, S. Density estimation using real NVP. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL https://openreview.net/forum?id=HkpbnH9lx.

Duane, S., Kennedy, A. D., Pendleton, B. J., and Roweth, D. Hybrid monte carlo. *Physics letters B*, 195(2):216–222, 1987.

Gabrié, M., Rotskoff, G. M., and Vanden-Eijnden, E. Adaptive monte carlo augmented with normalizing flows. *Proceedings of the National Academy of Sciences*, 119(10): e2109420119, 2022.

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. C., and Bengio, Y. Generative adversarial networks. *Commun. ACM*, 63(11):139–144, 2020. doi: 10.1145/3422622. URL https://doi.org/10.1145/3422622.

Gorham, J. and Mackey, L. W. Measuring sample quality with stein's method. In Cortes, C., Lawrence, N. D., Lee, D. D., Sugiyama, M., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pp. 226–234, 2015. URL https://proceedings.neurips.cc/paper/2015/hash/698d51a19d8a121ce581499d7b701668-Abstract.html.

Heek, J., Levskaya, A., Oliver, A., Ritter, M., Rondepierre, B., Steiner, A., and van Zee, M. Flax: A neural network library and ecosystem for JAX. 2023. URL http://github.com/google/flax.

Hirt, M., Titsias, M. K., and Dellaportas, P. Entropy-based adaptive hamiltonian monte carlo. In Ranzato, M., Beygelzimer, A., Dauphin, Y. N., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pp. 28482–28495, 2021.

Hoffman, M., Sountsov, P., Dillon, J. V., Langmore, I., Tran, D., and Vasudevan, S. Neutra-lizing bad geometry in hamiltonian monte carlo using neural transport. *arXiv preprint arXiv:1903.03704*, 2019.

Kingma, D. P. and Welling, M. Auto-encoding variational bayes. In Bengio, Y. and LeCun, Y. (eds.), *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014. URL http://arxiv.org/abs/1312.6114.

Kingma, D. P., Salimans, T., Jozefowicz, R., Chen, X., Sutskever, I., and Welling, M. Improved variational inference with inverse autoregressive flow. *Advances in neural information processing systems*, 29, 2016.

Levy, D., Hoffman, M. D., and Sohl-Dickstein, J. Generalizing hamiltonian monte carlo with neural networks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. URL https://openreview.net/forum?id=B1n8LexRZ.

Luengo, D., Martino, L., Bugallo, M., Elvira, V., and Särkkä, S. A survey of monte carlo methods for parameter estimation. *EURASIP Journal on Advances in Signal Processing*, 2020(1):1–62, 2020.

Neal, R. M. *MCMC using Hamiltonian dynamics*, volume 2. Chapman and Hall/CRC, 2011. URL https://www.mcmchandbook.net/index.html.

Neklyudov, K., Egorov, E., and Vetrov, D. P. The implicit metropolis-hastings algorithm. In Wallach, H. M., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E. B., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 13932–13942, 2019.

Neklyudov, K., Welling, M., Egorov, E., and Vetrov, D. P. Involutive MCMC: a unifying framework. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pp. 7273–7282. PMLR, 2020.

URL http://proceedings.mlr.press/v119/neklyudov20a.html.

Pasarica, C. and Gelman, A. Adaptively scaling the metropolis algorithm using expected squared jumped distance. *Statistica Sinica*, pp. 343–364, 2010.

Pinsker, M. S. *Information and information stability of random variables and processes*. Holden-Day, 1964.

Rezende, D. J. and Mohamed, S. Variational inference with normalizing flows. In Bach, F. R. and Blei, D. M. (eds.), *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pp. 1530–1538. JMLR.org, 2015. URL http://proceedings.mlr.press/v37/rezende15.html.

Rezende, D. J., Mohamed, S., and Wierstra, D. Stochastic backpropagation and approximate inference in deep generative models. In *International conference on machine learning*, pp. 1278–1286. PMLR, 2014.

Robert, C. P., Casella, G., and Casella, G. *Monte Carlo statistical methods*, volume 2. Springer New York, New York, NY, 2004. ISBN 978-1-4757-4145-2. doi: 10.1007/978-1-4757-4145-2_1. URL https://doi.org/10.1007/978-1-4757-4145-2_1.

Roberts, G. O. and Rosenthal, J. S. General state space Markov chains and MCMC algorithms. *Probability Surveys*, 1(none):20 – 71, 2004. doi: 10.1214/154957804100000024. URL https://doi.org/10.1214/154957804100000024.

Roberts, G. O. and Rosenthal, J. S. Examples of adaptive mcmc. *Journal of computational and graphical statistics*, 18(2):349–367, 2009.

Samsonov, S., Lagutin, E., Gabrié, M., Durmus, A., Naumov, A., and Moulines, E. Local-global mcmc kernels: the best of both worlds. *Advances in Neural Information Processing Systems*, 35:5178–5193, 2022.

Song, J., Zhao, S., and Ermon, S. A-NICE-MC: adversarial training for MCMC. In Guyon, I., von Luxburg, U., Bengio, S., Wallach, H. M., Fergus, R., Vishwanathan, S. V. N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pp. 5140–5150, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/2417dc8af8570f274e6775d4d60496da-Abstract.html.

Titsias, M. K. and Dellaportas, P. Gradient-based adaptive markov chain monte carlo. In Wallach, H. M., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E. B., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 15704–15713, 2019.

Valperga, R., Webster, K., Turaev, D., Klein, V., and Lamb, J. S. W. Learning reversible symplectic dynamics. In Firoozi, R., Mehr, N., Yel, E., Antonova, R., Bohg, J., Schwager, M., and Kochenderfer, M. J. (eds.), *Learning for Dynamics and Control Conference, L4DC 2022, 23-24 June 2022, Stanford University, Stanford, CA, USA*, volume 168 of *Proceedings of Machine Learning Research*, pp. 906–916. PMLR, 2022. URL https://proceedings.mlr.press/v168/valperga22a.html.

A. Architectures

All experiments are performed using compositions of parametric Hénon maps. Hénon maps are symplectic transformations on $\mathbb{R}^n \times \mathbb{R}^n$, $(x, y) \mapsto (\bar{x}, \bar{y})$, defined by

$$\begin{cases} \bar{x} = y + \eta \\ \bar{y} = -x + V(y), \end{cases} \tag{27}$$

with $V : \mathbb{R}^n \to \mathbb{R}^n$, and η a constant. The reason why we use Hénon maps, other than their approximation properties (see Valperga et al. (2022)), is because they are invertible analytically: for given V and η the inverse is simply

$$\begin{cases} x = -\bar{y} + V(\bar{x} - \eta) \\ y = \bar{x} - \eta. \end{cases} \tag{28}$$

For experiments with 2D distributions we use Hénon maps with the function V being a two-layer MLP with hidden dimension 32 and compose 5 of such layers to construct the function g_θ from Eq. (6). To sample from the Bayesian logistic regression posterior we set the hidden dimension of the two-layer MLP is 64.

For the discriminator we use the product parameterization using two three-layer MLP with hidden dimensions 32 for the experiments with the 2D distribution, and 128 for the Bayesian posterior.

The models have been trained on one NVIDIA A100. Training times are around 2 to 3 minutes for the simple distributions and about 5 to 10 minutes for the Bayesian posterior.

B. Analytic form of the densities

Following Song et al. (2017), for all experiments $p(v|x)$ is a Gaussian centered at zero with identity covariance. Now with $f(x|\mu, \sigma)$ denoting the log density of the Gaussian $\mathcal{N}(\mu, \sigma^2)$, the 2D log densities $U(x) = \log p(x)$ used in the experiments are

mog2:

$$U(x) = f(x|\mu_1, \sigma_1) + f(x|\mu_2, \sigma_2) - \log 2$$

where $\mu_1 = [5, 0]$, $\mu_2 = [-5, 0]$, $\sigma_1 = \sigma_2 = [0.5, 0.5]$.

mog6

$$U(x) = \sum_{i=1}^{6} f(x|\mu_i, \sigma_i) - \log 6$$

where $\mu_i = \begin{bmatrix} 5\sin\left(\frac{i\pi}{3}\right) \\ 5\cos\left(\frac{i\pi}{3}\right) \end{bmatrix}$ and $\sigma_i = [0.5, 0.5]$.

ring

$$U(x) = \left(\frac{\sqrt{x_1^2 + x_2^2} - 2}{0.32} \right)^2$$

ring5

$$U(x) = \min(u_1, u_2, u_3, u_4, u_5)$$

where $u_i = \left(\sqrt{x_1^2 + x_2^2} - i \right)^2 / 0.04$.

For the Bayesian logistic regression, we define the likelihood and prior as

$$p(\mathbf{y}|\mathbf{X}, \boldsymbol{\beta}) = \prod_{i=1}^{n} \left[\sigma(\mathbf{x}_i^T \boldsymbol{\beta}) \right]^{y_i} \left[1 - \sigma(\mathbf{x}_i^T \boldsymbol{\beta}) \right]^{1-y_i} \tag{29}$$

where $\sigma(z) = \frac{1}{1+e^{-z}}$

Then the unnormalized density of the posterior distribution for a dataset $D = \{\mathbf{X}, \mathbf{y}\}$ is

$$p(\boldsymbol{\beta}|\mathbf{y}, \mathbf{X}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) \propto p(\mathbf{y}|\mathbf{X}, \boldsymbol{\beta}) \cdot p(\boldsymbol{\beta}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) \tag{30}$$

where the Gaussian prior is $p(\boldsymbol{\beta}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \mathcal{N}(\boldsymbol{\beta}|\boldsymbol{\mu}, \boldsymbol{\Sigma})$ is a Gaussian with diagonal covariance.

We use three datasets: *german* (25 covariates, 1000 data points), *heart* (14 covariates, 532 data points) and *australian* (15 covariates, 690 data points). When computing the average log posterior we compute

$$\frac{1}{|D_{\text{test}}|} \sum_{x,y \in D_{\text{test}}} \log \frac{1}{|C|} \sum_{\theta \in C} p(y|x, \theta),$$

where C is a chain obtained with D_{train}.

C. Effective sample size

Following Song et al. (2017) given a chain $\{x_i\}_{i=1}^N$ we compute the ESS as:

$$\text{ESS}\left(\{x_i\}_{i=1}^N\right) = \frac{N}{1 + 2\sum_{s=1}^{N-1}\left(1 - \frac{s}{N}\right)\rho_s} \tag{31}$$

where ρ_s is the autocorrelation of x at lag s. We use the empirical estimate $\hat{\rho}_s$ of ρ_s:

$$\hat{\rho}_s = \frac{1}{\hat{\sigma}^2(N-s)} \sum_{n=s+1}^N (x_n - \hat{\mu})(x_{n-s} - \hat{\mu}) \tag{32}$$

where $\hat{\mu}$ and $\hat{\sigma}$ are the empirical mean and variance obtained by an independent sampler.

Following Song et al. (2017), we also truncate the sum over the autocorrelations when the autocorrelation goes below 0.05 to due to noise for large lags s.

D. Benchmarking time

We report in figure 6 the time it takes to perform one forward pass of the parametric proposal in the Ai-sampler compared to a single call to the gradient function of the Bayesian logistic regression posterior obtained with JAX autodifferentiation. We
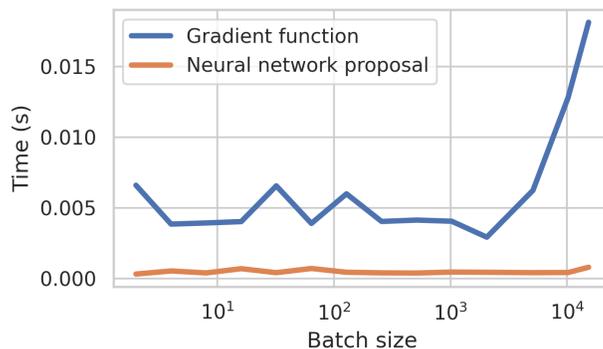

Figure 6. Time taken for a single call of the gradient function and neural network proposal vs. batch size.

do not compare running time with Song et al. (2017) as their implementation uses TensorFlow 1 which is not as efficient as JAX, which XLA to compile and run code on accelerators. We stress that time benchmark is to highlight the cost of multiple calls to the density gradient functions, especially in the case of complex Bayesian posterior distributions.
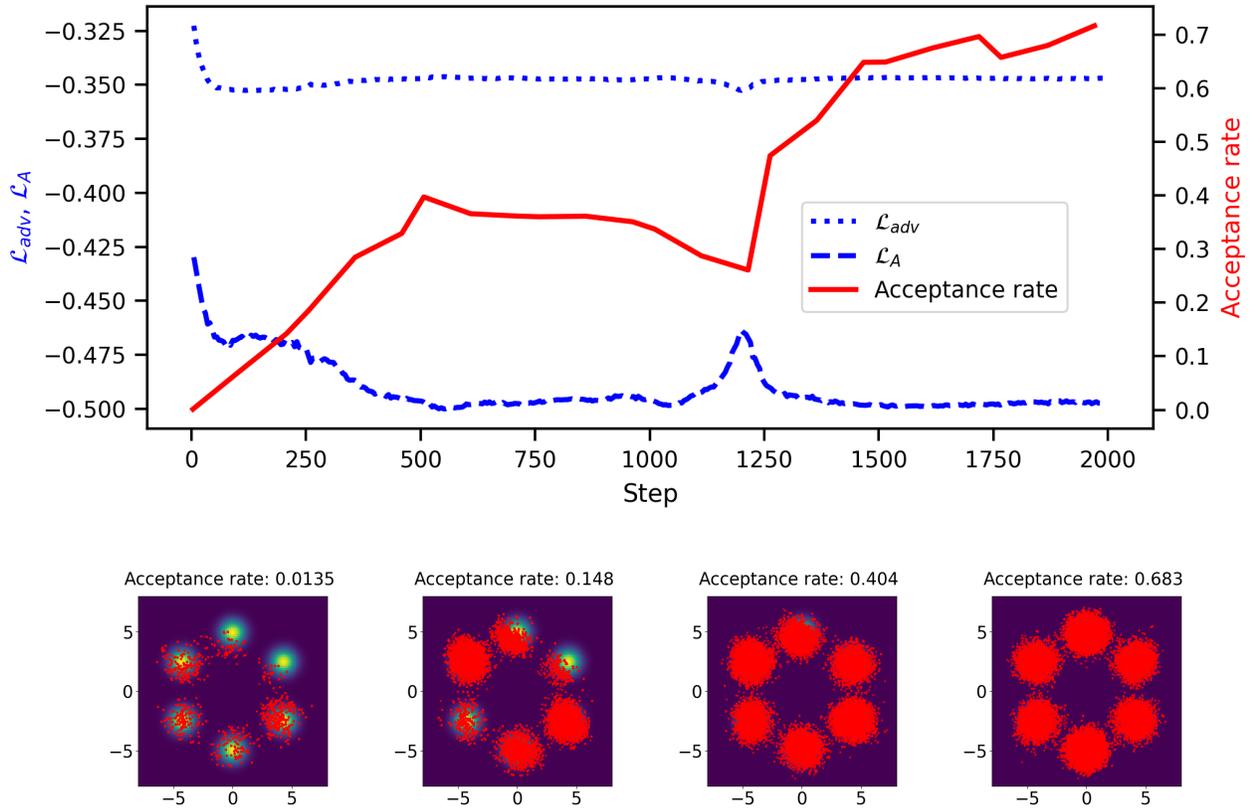
Figure 7. Adversarial objective and acceptance rate during training. Sample quality increasing during training.

E. Additional figures

In Fig. 7 we show an example of training curve, with the acceptance rate, during training.

F. Unbalanced Mixture of Gaussians

For unbalanced mixture of Gaussians with weights $(1/6, 1/6, 2/3)$ we provide total variation distance between ground truth and KDE estimation from chain samples. In order to asses mode capturing, we estimate mode weights from chain samples and provide KL difference between them and ground truth.
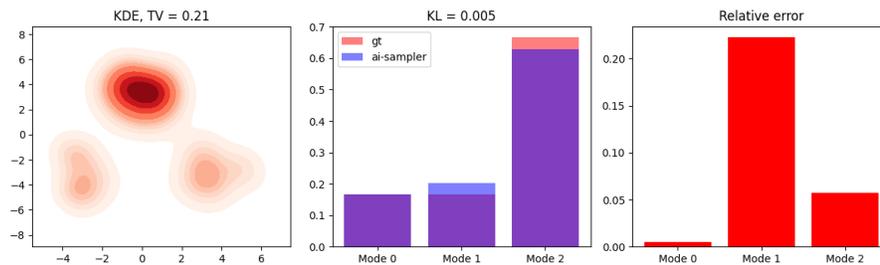

Figure 8. Unbalanced mixture of Gaussians.