A-NICE-MC

Adversarial Training for MCMC

Jiaming Song Shengjia Zhao Stefano Ermon

March 7, 2018

Stanford University

Table of contents

Motivation

Bayesian Inference

Parameters θ, observations $\mathcal{D}$:

Input prior $p(\theta)$ and likelihood $p(\mathcal{D}|\theta)$

Output posterior $p(\theta|\mathcal{D})$ through Bayes' rule:

$$p(\theta|\mathcal{D}) = \frac{p(\theta)p(\mathcal{D}|\theta)}{p(\mathcal{D})}$$

Problem: marginal $p(\mathcal{D})$ is intractable!

Solutions: *Variational Inference* and *Markov chain Monte Carlo*

Bayesian Inference

Variational Inference: approximate the posterior with some tractable model and minimize its distance with the posterior.

Examples	mean field approximation [2]
Advantages	optimization is efficient
Drawbacks	performance limited by choice of model

Bayesian Inference

Markov chain Monte Carlo: approximate the posterior with particles sampled from Markov chain with desired stationary distribution.

Method	*Proposal* for next particle + *Metropolis-Hastings*
Examples	Gibbs sampling [4], Hamiltonian Monte Carlo [8]
Advantages	reaches the true posterior asymptotically
Drawbacks	need many samples to obtain good estimates

Deep Bayesian Learning

Variational Inference <- Deep Learning: ✓

- stochastic gradient descent as optimization algorithm
- expressive function approximations to represent model

Markov chain Monte Carlo <- Deep Learning: ✗

- proposals are hand-designed in general
- cannot apply expressive function approximations directly
- hard to evaluate / optimize metrics

Outline

We introduce A-NICE-MC, a new method for *training* flexible MCMC kernels.

- proposals are parameterized using (deep) neural networks
- use adversarial methods to train a Markov chain that
  - matches a target stationary quickly (burn-in)
  - achieves low autocorrelation between samples (mixing)
- learned proposals are much more efficient than traditional ones

Markov chain Monte Carlo + Deep Learning: ✓

Notations and Problem Setup

Notations

A sequence of *continuous* random variables $\{x_t\}_{t=0}^{\infty}$ is drawn through the following Markov chain:

$$x_0 \sim \pi^0 \qquad x_{t+1} \sim T_\theta(x_{t+1}|x_t)$$

where

- $T_\theta(\cdot|x)$: a stochastic transition kernel parametrized by θ
- π^0: some initial distribution for x_0.
- π_θ^t: state distribution at time t.

T_θ is defined through an *implicit generative model* $f_\theta(\cdot|x, v)$, where $v \sim p(v)$ is an auxiliary random variable.

Problem Setup

Let $p_d(x)$ be a target distribution over $x \in \mathbb{R}^n$, e.g.:

- a data distribution (which we can sample from)
- an (intractable) posterior distribution

Our objective is to find a T_θ such that:

1. **Low bias:** The stationary distribution is close to the target distribution (minimize $|\pi_\theta - p_d|$).
2. **Efficiency:** $\{\pi_\theta^t\}_{t=0}^\infty$ converges quickly (minimize t such that $|\pi_\theta^t - p_d| < \delta$).
3. **Low variance:** Samples from one chain $\{x_t\}_{t=0}^\infty$ should be as uncorrelated as possible (minimize autocorrelation of $\{x_t\}_{t=0}^\infty$).

Settings

Problem setup:

Input A target distribution $p_d(x)$

Output A transition kernel $T_\theta(\cdot|x)$

We consider two settings for specifying the target distribution.

- $p_d(x)$ is a data distribution (samples, no analytic expression)
- $p_d(x)$ an analytic expression (up to normalization constant, no samples)

Adversarial Training for Markov Chains

Parametrized Markov Chains

Assume we have direct access to samples from $p_d(x)$, and the transition kernel $T_\theta(x_{t+1}|x_t)$ is the following implicit generative model:

$$v \sim p(v) \quad x_{t+1} = f_\theta(x_t, v) \tag{1}$$

for which the stationary $\pi_\theta(x)$ exists.

Goal: find θ such that $\pi_\theta(x)$ is close to p_d.

Training Markov Chains

Likelihood-based Approaches:

- the value of $\pi_\theta(x)$ is typically intractable to compute
- the marginal distribution $\pi_\theta^t(x)$ at time t is also intractable (integration over all the possible paths)

Likelihood-free Apporoaches

- sampling is easy for Markov chains!
- likelihood-free methods only requires samples
- Example: Generative Adversarial Networks [5]

Generative Adversarial Networks

Generator $G(z)$: generates samples by transforming a noise variable $z \sim p(z)$ into $G(z)$

Discriminator $D(x)$: trained to distinguish between samples from the generator and samples from p_d.

This describes the following objective [1]:

$$\min_G \max_D V(D, G) = \min_G \max_D \mathbb{E}_{x \sim p_d}[D(x)] - \mathbb{E}_{z \sim p(z)}[D(G(z))] \tag{2}$$

Likelihood-free Training for Markov Chains

In our settings:

- $p_d(x)$ is the empirical distribution from the samples ✓
- $G_\theta(z)$ is the stationary? approximate with state after t steps? ✗

It is hard to sample from the stationary or optimize through a long chain!

Conditions for Stationary Distribution

We consider two necessary conditions for p_d to be a stationary:

- p_d should be close to π_b for some time step b
- p_d is a fixed point for the transition operator

We can construct an objective that can be optimized efficiently through the two conditions.

Markov GAN

Markov GAN (MGAN) objective:

$$\min_{\theta} \max_{D} \mathbb{E}_{x \sim p_d}[D(x)] - \lambda \mathbb{E}_{\bar{x} \sim \pi_\theta^b}[D(\bar{x})] - (1-\lambda)\mathbb{E}_{x_d \sim p_d, \bar{x} \sim T_\theta^m(\bar{x}|x_d)}[D(\bar{x})] \quad (3)$$

where

- $\lambda \in (0,1), b \in \mathbb{N}^+, m \in \mathbb{N}^+$ are hyperparameters
- $\bar{x}$ denotes "fake" samples from the generator
- $T_\theta^m(x|x_d)$ denotes the distribution of x when the transition kernel is applied m times, starting from some "real" sample x_d

Markov GAN

Markov GAN (MGAN) objective:

$$\min_{\theta} \max_{D} \mathbb{E}_{x \sim p_d}[D(x)] - \lambda \underbrace{\mathbb{E}_{\bar{x} \sim \pi_\theta^b}[D(\bar{x})]}_{\text{converge to } p_d} - (1-\lambda) \underbrace{\mathbb{E}_{x_d \sim p_d, \bar{x} \sim T_\theta^m(\bar{x}|x_d)}[D(\bar{x})]}_{\text{fixed point at } p_d} \quad (4)$$

We use two types of samples from the generator for training:

1. Samples after b transitions, starting from $x_0 \sim \pi^0$.
2. Samples after m transitions, starting from $x_d \sim p_d$.

Justifications

Proposition

Consider a sequence of ergodic Markov chains over state space $\mathcal{S}$. Define π_n as the stationary distribution for the n-th Markov chain, and π_n^t as the probability distribution at time step t for the n-th chain. If the following two conditions hold:

1. $\exists b > 0$ *such that the sequence* $\{\pi_n^b\}_{n=1}^{\infty}$ *converges to* p_d *in total variation;*
2. $\exists \epsilon > 0$, $\rho < 1$ *such that* $\exists M > 0, \forall m > M$ *if* $\|\pi_m^t - p_d\|_{TV} < \epsilon$, *then* $\|\pi_m^{t+1} - p_d\|_{TV} < \rho \|\pi_m^t - p_d\|_{TV}$;

then the sequence of stationary distributions $\{\pi_n\}_{n=1}^{\infty}$ *converges to* p_d *in total variation.*

Sketch of Proof

Proof.

The goal is to prove that $\forall \delta > 0, \exists N > 0, T > 0$, such that $\forall n > N, t > T, \|\pi_n^t - p_d\|_{\mathrm{TV}} < \delta$.

- $\exists N > 0$, such that $\forall n > N, \|\pi_n^b - p_d\|_{\mathrm{TV}} < \epsilon$ (Assumption 1).
- $\forall n > \max(N, M), \forall \delta > 0, \exists T = b + \max(0, \lceil \log_\rho \delta - \log_\rho \epsilon \rceil) + 1$, such that $\forall t > T, \|\pi_n^t - p_d\|_{\mathrm{TV}} < \delta$ (Assumption 2).

Hence the sequence $\{\pi_n\}_{n=1}^{\infty}$ converges to p_d in total variation. $\square$

Example: Generative Model for Images

We experiment with a distribution p_d over images, such as digits (MNIST) and faces (CelebA), where $x_{t+1} = f_\theta(x_t, v)$ is defined as

$$z = \text{encoder}_\theta(x_t) \quad z' = \text{ReLU}(z + \beta v) \quad x_{t+1} = \text{decoder}_\theta(z') \tag{5}$$

where β is a hyperparameter we set to 0.1.

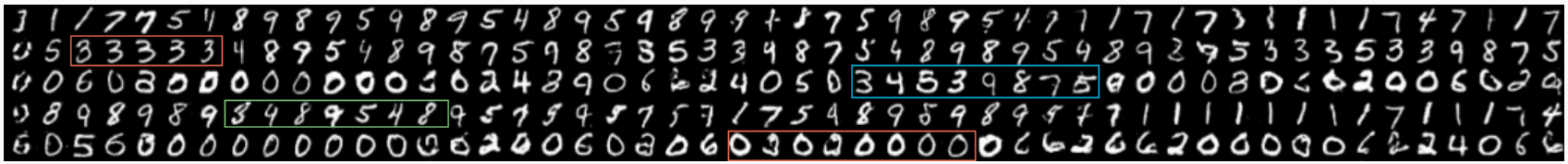


Figure 1: Visualizing samples of π_1 to π_{50} (each row) from a model trained on the MNIST dataset. Consecutive samples can be related in label (red box), inclination (green box) or width (blue box).

Transition Probabilities on MNIST

We use a classifier to classify the generated images and evaluate the class transition probabilities $T_\theta(y_{t+1}|y_t)$

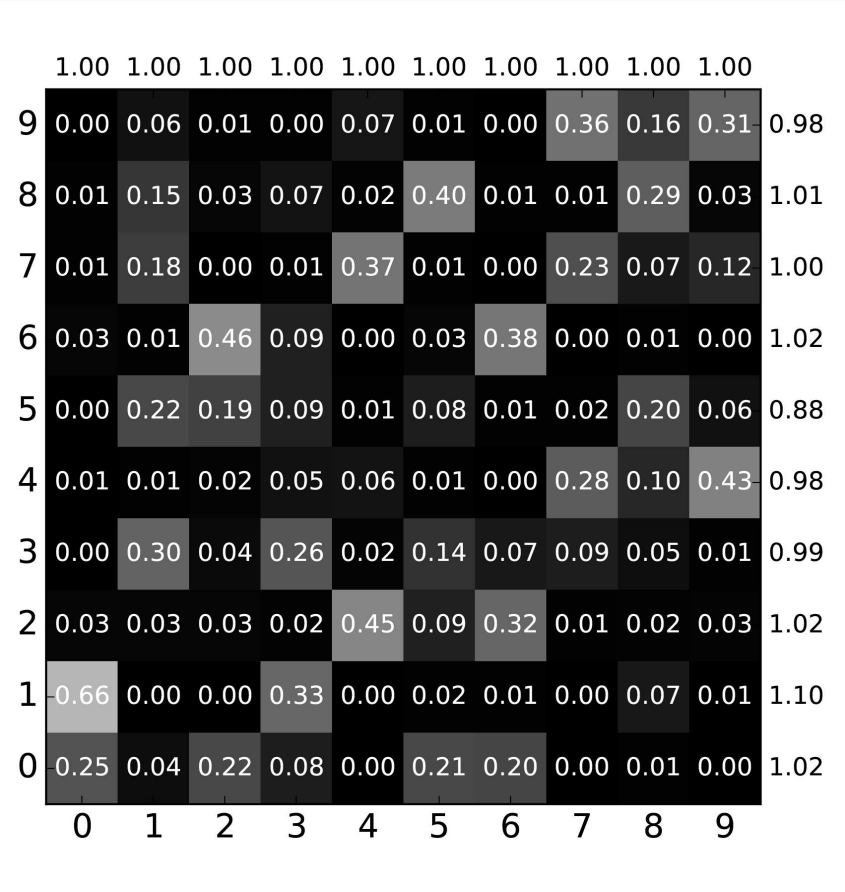


Figure 2: The transition is not symmetric!

Adversarial Training for MCMC

Analytical Target

Now consider the settings where the target distribution p_d is specified by an analytical expression:

$$p_d(x) \propto \exp(-U(x)) \tag{6}$$

where

- $U(x)$ is a known energy function
- normalization constant for $U(x)$ is not available

There are two additional challenges:

- We want the stationary to be *exactly* p_d
- We do not have direct access to samples from p_d

Metropolis Hastings

We use ideas from the Markov Chain Monte Carlo (MCMC) literature to address the first challenge.

Detailed Balance: $p_d(x)T_\theta(x'|x) = p_d(x')T_\theta(x|x')$ for all x and x'.

Metropolis-Hastings

- a sample x' is first obtained from a *proposal distribution* $g_\theta(x'|x)$
- x' is accepted with the following probability:

$$A_\theta(x'|x) = \min\left(1, \exp(U(x) - U(x'))\frac{g_\theta(x|x')}{g_\theta(x'|x)}\right) \tag{7}$$

Let $T_\theta(x'|x) = g_\theta(x'|x)A_\theta(x'|x)$, then the Markov chain has stationary of p_d [6].

Challenges

Performance depends heavily on the choice of the proposal.

What should we choose? Recall our desiderata:

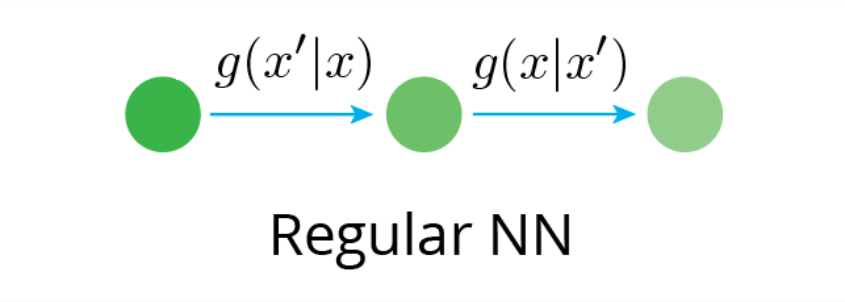
1. **Low bias:** The stationary distribution is close to the target distribution. (always true due to Metropolis-Hastings)
2. **Efficiency:** $\{\pi_\theta^t\}_{t=0}^\infty$ converges quickly. (need reasonable acceptance rate, T_θ not longer differentiable)
3. **Low variance:** Samples from one chain $\{x_t\}_{t=0}^\infty$ should be as uncorrelated as possible (haven't discussed low autocorrelation in MGAN).

Challenge I: Low acceptance rate

Low acceptance if we use an implicit generative model directly:

- $g_\theta(x'|x)$ is high
- $g_\theta(x|x')$ is low

So $g_\theta(x|x')/g_\theta(x'|x)$ is low.



Challenge II: Training

Kernel is non-differentiable (cannot optimize like a recurrent net).

Score function gradient estimator (like REINFORCE) also fails!

- High variance in gradient estimates
- Low acceptance rates: MH tend to reject very frequently (99.9%)

Our Approach

We address the challenges through:

- Introduce a *NICE proposal*, which avoids low acceptance rates
- Train the *NICE proposal* (in an adversarial fashion) that is end-to-end differentiable
- Propose a method that targets low autocorrelation

We call the approach *Adversarial NICE Monte Carlo* (A-NICE-MC).

Non-linear Independent Component Estimation (NICE)

Flow models: generative models for x through a bijection $f : h \to x$

- $x \in \mathbb{R}^n$, $h \in \mathbb{R}^n$
- h has a fixed prior $p_H(h)$
- $p_X(x) = p_H(f^{-1}(x)) \left| \det \frac{\partial f^{-1}(x)}{\partial x} \right|^{-1}$

Non-linear Independent Component Estimation (NICE)

Flow models: generative models for x through a bijection $f : h \to x$

NICE: a *volume preserving* flow model

- **Volume preserving**: $|\det \frac{\partial f(h)}{\partial h}| = |\det \frac{\partial f^{-1}(x)}{\partial x}| = 1$
- Constructed by stacking **additive coupling layers**, mappings from (y, z) to (y', z')

$$y' = y \qquad z' = z + m(y) \tag{8}$$

  where $m(\cdot)$ is a neural network.

With a NICE f_θ, it is easy to obtain f_θ^{-1}!

A NICE Proposal

Our proposal considers a NICE model $f_\theta(x, v)$ with its inverse f_θ^{-1}, where $v \sim p(v)$ is the auxiliary variable. We draw a sample x' from the proposal $g_\theta(x', v'|x, v)$ using the following procedure:

1. Randomly sample $v \sim p(v)$ and $u \sim \text{Uniform}[0, 1]$;
2. If $u > 0.5$, then $(x', v') = f_\theta(x, v)$;
3. If $u \leq 0.5$, then $(x', v') = f_\theta^{-1}(x, v)$.

We call this proposal a *NICE proposal*.

A NICE Property

Theorem

For any (x, v) *and* (x', v') *in their domain, a NICE proposal* g_θ *satisfies*

$$g_\theta(x', v'|x, v) = g_\theta(x, v|x', v')$$

Proof.

For any (x, v) and (x', v')

$$\begin{aligned} g(x', v'|x, v) &= \frac{1}{2}\mathbb{I}(x', v' = f(x, v)) + \frac{1}{2}\mathbb{I}(x', v' = f^{-1}(x, v)) \\ &= \frac{1}{2}\mathbb{I}(x, v = f^{-1}(x', v')) + \frac{1}{2}\mathbb{I}(x, v = f(x', v')) \\ &= g(x, v|x', v') \end{aligned} \tag{9}$$

where $\mathbb{I}(\cdot)$ is the indicator function. □

End-to-end Differentiable Training

Use the MGAN objective with f as transition kernel!

- Ignore f^{-1} and the MH step during training
- Use them only during MCMC inference.

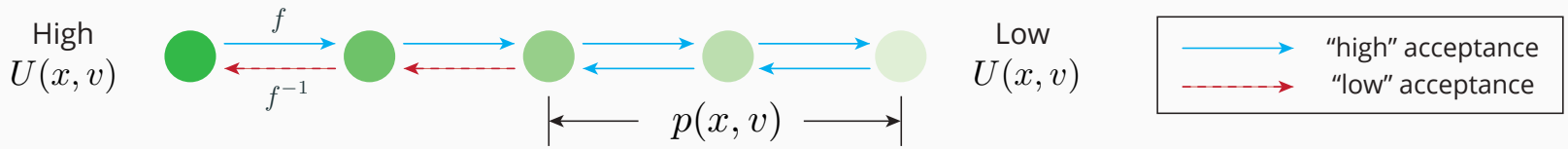


Figure 3: Sampling process of A-NICE-MC. Each step, the proposal executes f_θ or f_θ^{-1}. Outside the high probability regions f_θ will guide x towards $p_d(x)$, while MH will tend to reject f_θ^{-1}. Inside high probability regions both operations will have a reasonable probability of being accepted.

Pairwise Discriminator

Low variance: Samples from one chain $\{x_t\}_{t=0}^{\infty}$ should be as uncorrelated as possible (low autocorrelation).

Effective sample size: an important measurement for MCMC performance

- Let $V = \mathrm{Var}_q[\sum_{i=1}^{N} x_i/N]$ be the variance of the mean estimate through the MCMC samples.
- $ESS(\{x_i\}_1^N)$ is the number of independent samples from $p(x)$ needed in order to achieve the same variance, i.e. $\mathrm{Var}_p[\sum_{j=1}^{M} x_j/M] = V$

Pairwise Discriminator

Low variance: Samples from one chain $\{x_t\}_{t=0}^{\infty}$ should be as uncorrelated as possible (low autocorrelation).

Effective sample size: an important measurement for MCMC performance

$$ESS(\{x_i\}_1^N) = \frac{N}{1 + 2\sum_{s=1}^{N-1}(1 - \frac{s}{N})\rho_s} \tag{10}$$

where ρ_s denotes the autocorrelation under q of x at lag s (lower ρ_s gives higher ESS).

Unfortunately, MGAN does not optimize for autocorrelation!

Pairwise Discriminator

A simple trick to train for low autocorrelation / high ESS.

Instead of taking one sample, the discriminator takes *a pair of samples* (x_1, x_2)

Real data : a pair of independent samples from p_d

Generated data : a pair of correlated samples from the chain

- $x_1 \sim p_d$, or $x_1 \sim \pi_\theta^b$.
- $x_2 \sim T_\theta^m(\cdot|x_1)$

Match the distribution of correlated *generated* samples to the distribution of independent *data* samples!

Bootstrap

We need samples from p_d for likelihood-free training:

- with (almost) any θ, MCMC with proposal g_θ has stationary p_d
- this is an unbiased way to obtain samples

Consider the following bootstrap procedure

1. Initialize θ randomly
2. Obtain samples $\{x_i\}_{i=1}^N$ through MCMC with g_θ as proposal
3. Train g_θ with pairwise discriminator
4. Go to 2 and repeat.

Experiments

Settings

Two settings:

- Synthetic 2D energy functions
- Bayesian logistic regression

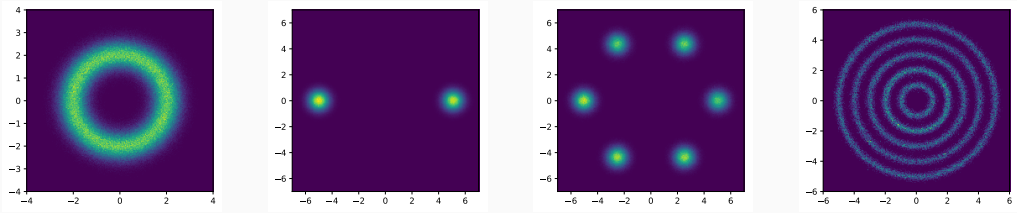


Figure 4: Densities of **ring**, **mog2**, **mog6** and **ring5** (from left to right).

Methods

We consider MCMC on continuous random variables, where $U(x)$ is differentiable.

- **A-NICE-MC**: proposal based on NICE
- **Hamiltonian Monte Carlo**: proposal based on Hamiltonian dynamics

We consider three measurements:

- **ESS** (for fixed number of samples from Markov chain)
- **ESS per second** (the measurement we care about in practice)
- Mean absolute error for estimating statistics.

Hyperparameters

A-NICE-MC: we use the same hyperparameters for all 2D tasks, and same for all Bayesian LR tasks.

HMC: we tune for the best hyperparameter

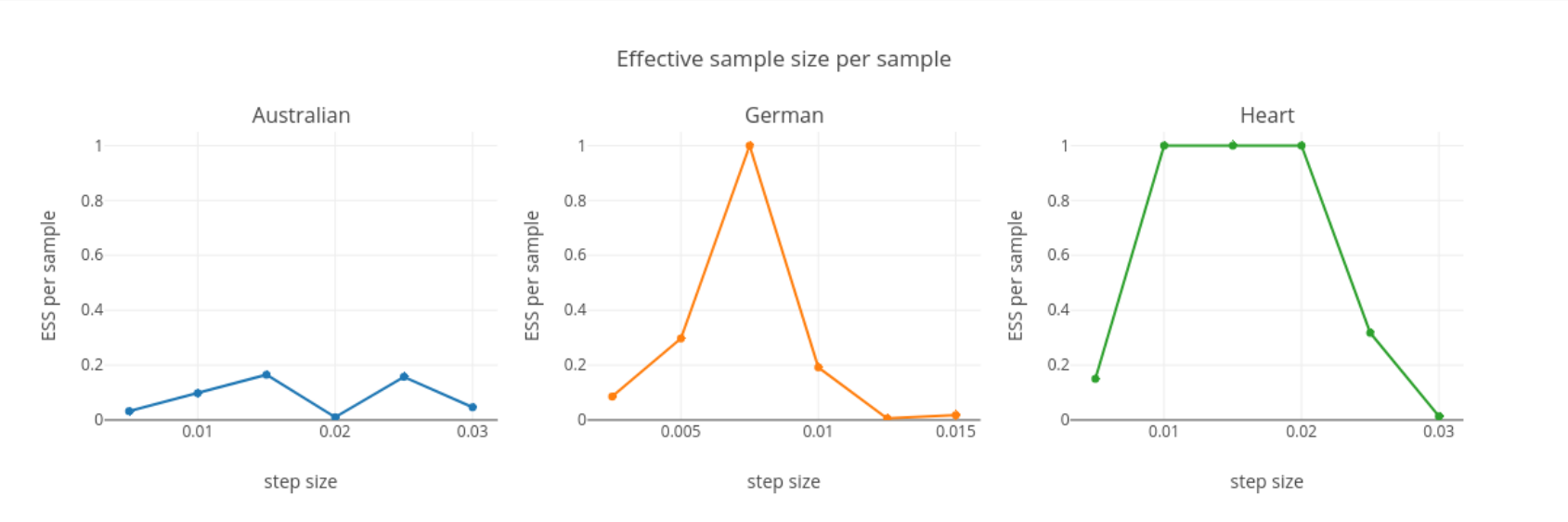


Figure 5: HMC is sensitive to changes in hyperparemeter.

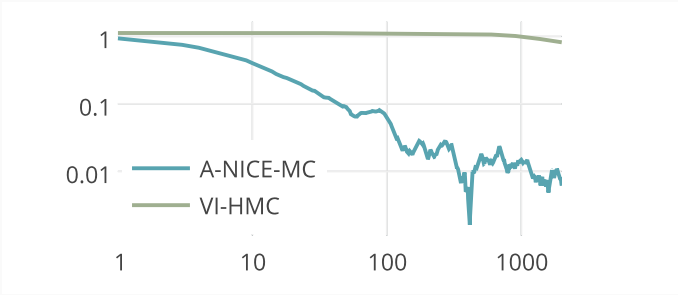
Synthetic Energy Functions

Around 100x improvement in ESS/s.

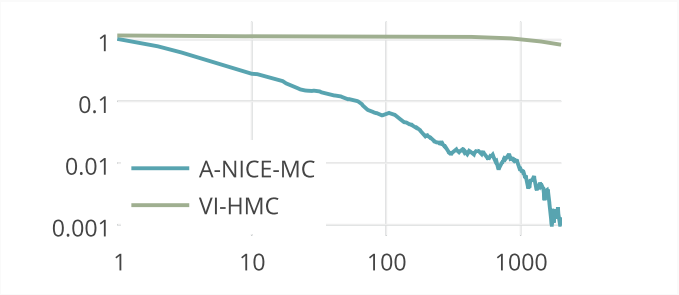
Table 1: Performance of MCMC samplers as measured by Effective Sample Size (ESS). Higher is better (1000 maximum). Averaged over 5 runs under different initializations.

ESS	A-NICE-MC	HMC	ESS/s	A-NICE-MC	HMC
ring	**1000.00**	**1000.00**	ring	**128205**	**121212**
mog2	**355.39**	1.00	mog2	**50409**	78
mog6	**320.03**	1.00	mog6	**40768**	39
ring5	**155.57**	0.43	ring5	**19325**	29

Estimating Statistics on ring5



(a) $\mathbb{E}[\sqrt{x_1^2 + x_2^2}]$



(b) $\mathrm{Std}[\sqrt{x_1^2 + x_2^2}]$

Figure 6: Mean absolute error for estimating the statistics in *ring5* w.r.t. simulation length. Averaged over 100 chains.

Does Training Improve ESS?

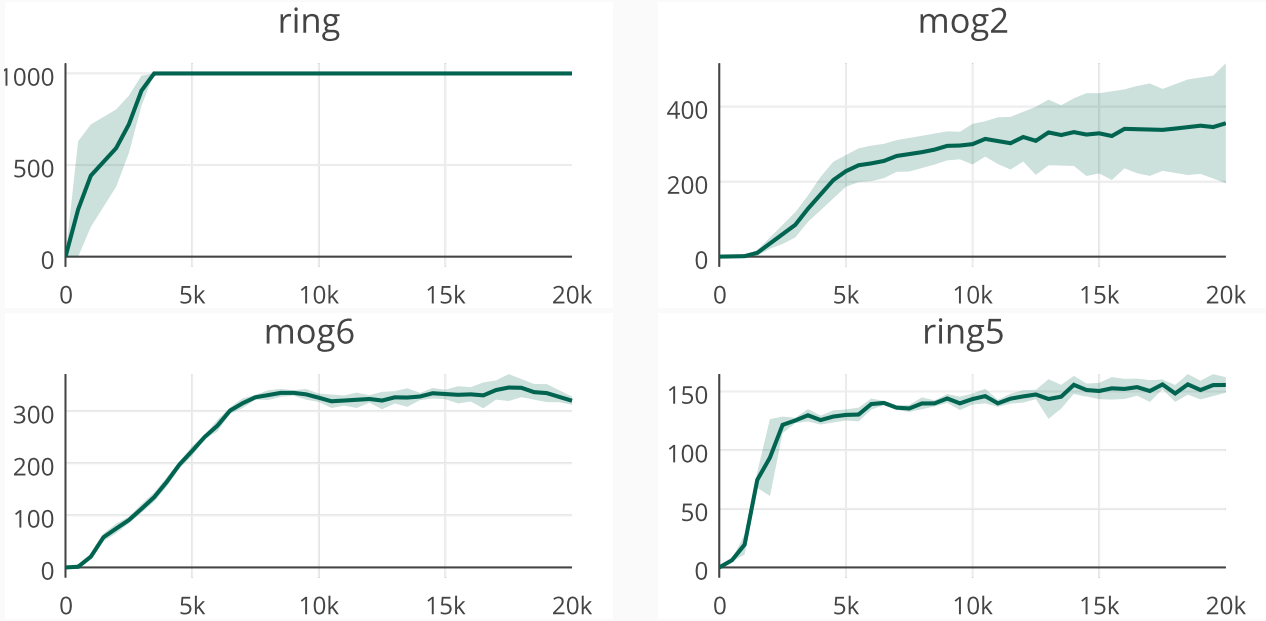


Figure 7: ESS with respect to the number of training iterations.

Admittedly, training introduces an additional computational cost which HMC could utilize to obtain more samples initially (not taking parameter tuning into account), yet the initial cost can be amortized thanks to the improved ESS.

Bayesian Logistic Regression

A more realistic problem where HMC is a very strong baseline.

Querying $U(x)$ or $\nabla U(x)$ is equivalent to a pass through the dataset

- HMC uses many $\nabla U(x)$ queries for one proposal
- A-NICE-MC only does a forward pass through f (or f^{-1})

In general, A-NICE-MC proposals are much cheaper to run than HMC ones!

Bayesian Logistic Regression

Table 3: ESS and ESS per second for Bayesian logistic regression tasks.

ESS	A-NICE-MC	HMC
german	926.49	**2178.00**
heart	1251.16	**5000.00**
australian	1015.75	**1345.82**

ESS/s	A-NICE-MC	HMC
german	**1289.03**	216.17
heart	**3204.00**	1005.03
australian	**1857.37**	289.11

3-9x improvement in terms of ESS/s.

Summary

We introduce A-NICE-MC, a likelihood-free method for *training* flexible MCMC kernels, which

- constructs proposals with NICE, a volume preserving flow
- uses likelihood-free methods for efficient end-to-end training
- matches a target stationary quickly (good burn-in)
- encourages low autocorrelation between samples (good mixing)
- achieves significant empirical (ESS/s) improvements over HMC.

Code https://github.com/ermongroup/a-nice-mc

Paper https://arxiv.org/abs/1706.07561

Questions?

Animation

HMC

A-NICE-MC

Existing Problems

- Not training directly on the chain that we care about
- Obtaining samples is much harder in high-dimensional regions
- Exploration (potentially visit more modes) vs exploitation (training over existing sampled data)
- Training efficiency
- Incorporate $\nabla U(x)$ (see [7] for a follow-up on this work)

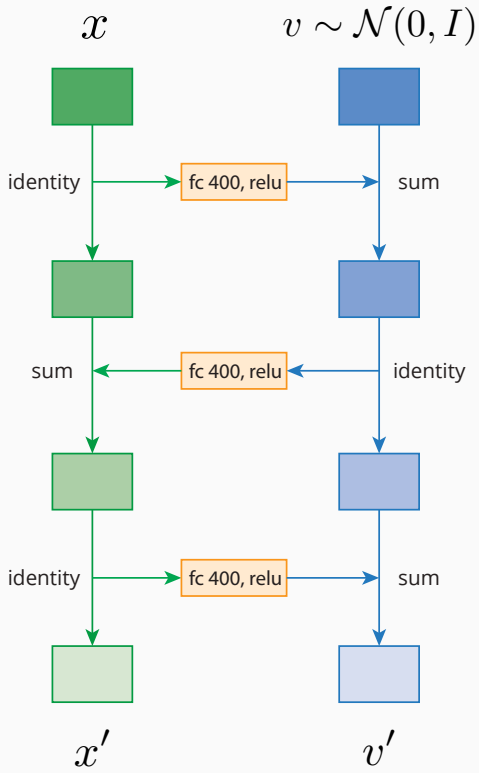
Gelman's R hat diagnostic [3]

Evaluates performance across multiple sampled chains.

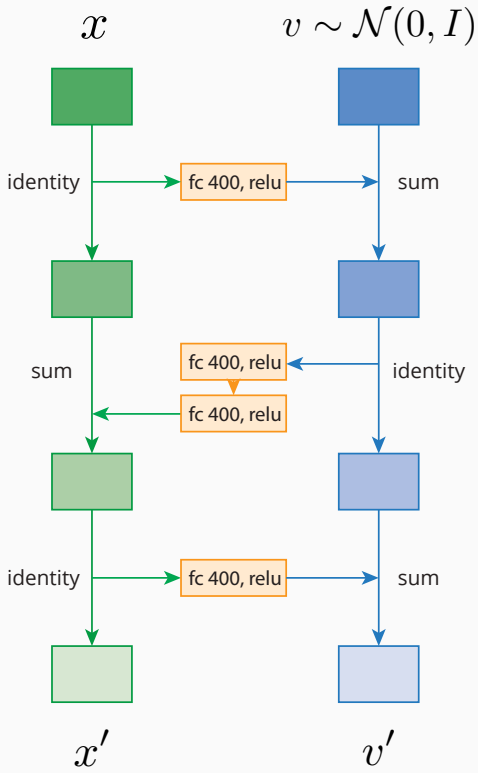
The perfect value is 1, and 1.1-1.2 would be regarded as too high.

- HMC gives a R hat value of 1.26 in *ring5*
- A-NICE-MC gives a R hat value of 1.002 in *ring5*

Architecture for A-NICE-MC



(a) NICE architecture for energy functions.



(b) NICE architecture for Bayesian logistic regression.

References i


M. Arjovsky, S. Chintala, and L. Bottou.
Wasserstein gan.
arXiv preprint arXiv:1701.07875, 2017.


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


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

References ii


A. E. Gelfand and A. F. Smith.
Sampling-based approaches to calculating marginal densities.
Journal of the American statistical association, 85(410):398–409, 1990.


I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio.
Generative adversarial nets.
In *Advances in neural information processing systems*, pages 2672–2680, 2014.


W. K. Hastings.
Monte carlo sampling methods using markov chains and their applications.
Biometrika, 57(1):97–109, 1970.

References iii


D. Levy, M. D. Hoffman, and J. Sohl-Dickstein.
Generalizing hamiltonian monte carlo with neural networks.
arXiv preprint arXiv:1711.09268, 2017.


R. M. Neal et al.
Mcmc using hamiltonian dynamics.
Handbook of Markov Chain Monte Carlo, 2:113–162, 2011.