Normalizing Flows for Proposal Distributions in MCMC

Leonardo Micheli Belo

july 2025

Abstract

Normalizing flows (NF) offer a principled way to turn a simple latent distribution into an expressive density through a sequence of invertible, differentiable maps. Recent work shows that combining NFs with Markov-chain Monte-Carlo (MCMC) can accelerate convergence, especially in multimodal or high-dimensional targets. We integrate a RealNVP flow with a Metropolis-within-Gibbs sampler: the chain evolves in the latent space $\mathcal{Z}$ while the flow transports proposals to the data space $\mathcal{X}$. Empirically, the method yields an order-of-magnitude improvement in effective sample size over a random-walk Metropolis baseline on a ten-dimensional, tri-modal Gaussian mixture as done in the experiments, demonstrating that NF-guided proposals can improve sampling efficiency.

1 Introduction

Markov chain Monte Carlo (MCMC) algorithms such as Metropolis–Hastings (MH) and Gibbs sampling are the base of modern Bayesian inference, which has to deal with many times with intractable integration. Yet in high dimension or in the presence of multiple separated modes, the random-walk proposals commonly used in MH mix slowly, leading to low effective sample size (ESS) and long run-times.

Normalizing flows (NFs), as described in Papamakarios et al., have emerged as flexible generative models capable of approximating complex continuous densities while retaining tractable likelihoods. Because they are invertible, an NF can transport a simple proposal distribution into one that closely matches the target, potentially reducing random-walk waste.

In this work we use a RealNVP flow onto an MH Gibbs kernel. Our objectives are to search the use the NF to warp the latent proposal so that steps in z translate to large, well-aligned moves in x; scalability and simplicity, which in our case is to rely on lightweight RealNVP layers and a single adaptive phase during burn-in.

2 Background

2.1 Metropolis Hastings

Given a target density $\pi(x)$ known up to a constant, MH constructs a Markov chain $x^{(t)}$ with proposal $q(.|x^{(t)})$. A candidate x' is accepted with probability $\alpha = min(1, \frac{\pi(x')q(x^{(t)}|x')}{\pi(x^{(t)})q(x'|x^{(t)})})$.

The acceptance rule enforces the **detailed balance** condition $\pi(x)P(x, dx') = \pi(x')P(x', dx)$, implying that π is stationary. Under minor irreducibility conditions the chain is also positive Harris recurrent, ensuring consistency of ergodic averages. A common choice is an isotropic Gaussian proposal $q(x' \mid x) = \mathcal{N}(x, \sigma^2 I)$. In d dimensions the optimal scale is $\sigma \propto d^{-1/2}$, yielding an asymptotically optimal acceptance rate of 0.234. Consequently the integrated autocorrelation time grows $\mathcal{O}(d)$, which motivates more sophisticated proposals such as the one introduced in this paper.

2.2 Gibbs Sampling

Gibbs replaces the full-dimensional proposal by sequentially drawing from conditional distributions, often updating one or a few coordinates at a time. Assume $x = (x_1, \dots, x_d)$ and that each full conditional $\pi(x_j \mid x_{-j})$ is tractable. Gibbs sampling iteratively updates one (or more) coordinates from their conditionals. For $j = 1, \dots, d$ draw

$$x_j^{(t+1)} \;\sim\; \pi\big(\cdot \mid x_{-j}^{(t+1)}\big), \tag{1}$$

where $x_{-j}^{(t+1)}$ denotes the most recent values of the other coordinates. Because proposals are always accepted, Gibbs often mixes faster than random–walk MH when strong conditional structure is present.

The resulting *Metropolis-within-Gibbs* variant keeps the acceptance formula but in a low-dimensional sub-space, improving local acceptance rates. When a block conditional is not available in closed form, one can embed an MH sub–kernel for that block, preserving global detailed balance.

2.3 Normalizing Flow

Normalizing flow is a type of technique of generative modeling, which can be thought of as inverting an inference process. In the normalizing flow case, the process is invertible, then we can transform the data into a tractable distribution. Normalizing flows provide a general way of constructing flexible probability distributions over continuous random variables.

The main idea is to express x (the original data) as a transformation T of a real vector $z \in \Re^D$ (same dimension of $\mathcal{X}$) sampled from $p_Z(z)$ as $x = T(z)$, which T and p_Z have parameters of their own ϕ and ψ respectively and T has inverse T^{-1} both differentiable, characterizing a diffeomorphism. Under these conditions, we can write:

$$p_x(x) = p_u(T^{-1}(x))|det\, J_{T^{-1}}(x)|, \tag{2}$$

where $J_T(u)$ is the Jacobian $D \times D$ matrix of all partial derivatives of T.

A consequence of T be invertible and differentiable transformation is that can be composable. That way, we can build complex tranformations by composing multiple instances of simpler transformations, i.e. create a chain of multiple transformations $T_1, ..., T_K$ to obtain $T = T_K \circ ... \circ T_1$, where each T_k transforms z_{k-1} into z_k, assuming $z_0 = u$ and $z_K = x$. The application of this property is what we refer as 'flow', where the trajectory is a collection of samples from $p_u(u)$ follow as they are gradually transformed by the sequence of transformations $T_1, ..., T_K$.

3 Algorithm

3.1 Related works

There are previous works to integrate Normalizing flows and MCMC with different approach. Let's review some of them:

- **Gflow MC** (Schönle et. al.) [5]: It's a method to use normalizing flow to create a diffeomorphism transformation from data space to latent space (chosen to be simple). So, the algorithm use a Metropolis-within-Gibbs in the latent space and we achieve the distribution in data space thanks to the transformation of normalizing flow. In prior works using normalizing flows and MCMC, tthey only move locally in the latent space. If the latent distribution still has complex structure (like multiple modes far apart), local moves might not mix well (slow convergence, poor exploration), so the GflowMC introduces partial updates in the latent space instead of updating the whole latent vector at once, what increased efficiency in high dimensions. However, there are still some problems such as flow expressiveness (depends of a powerful NF), slow exploration (especially when the variables are highly correlated) and curse of dimensionality (In very high-dimensional spaces, updating one or a few coordinates at a time can require many iterations to fully explore the space).
- **Markovian Flow Matching** (Cabezas et. al.) [6] try to accelerate MCMC integrating with a flow trained with flow matching. In this method, a Continuous normalizing flow (CNF) is trained via flow matching to learn a smooth, invertible transformation between a simple reference distribution (like a Gaussian) and the target distribution. During sampling, the CNF is used to generate efficient proposal moves for the MCMC algorithm in the way that samples are either mapped through the CNF or proposed directly in the transformed (latent) space and then mapped back. However, there are still some problems such as flow expressiveness, computational cost and memory usage.

3.2 The model: NF-guided Metropolis-within-Gibbs

The Normalizing flow will be use as guide proposal. All start with a base distribution $z \sim N(0, I)$, we place a RealNVP flow $f_\phi : \mathcal{Z} \to \mathcal{X}$, both in $\Re^d$, so $x = f_\phi(z)$. As f_ϕ is a diffeomorphism, we can obtain $\tilde{\pi}(z) = \pi(f_\phi(z))|detJ_\phi(z)|$, where $\pi(z) = \tilde{\pi}(z)/Z$, so we can evaluate $log\tilde{\pi}(z)$ with one forward pass.

RealNVP is a normalizing flow model designed for generative modeling. The key idea is to transform a simple distribution (like a standard Gaussian) into a complex data distribution using a sequence of invertible, differentiable functions.

The proposal mechanism works as follow at iteration t:

- choose a random block S_t of size $k \approx d$;
- set $z'_{S_t} = z^{(t)}_{S_t} + \sigma\epsilon, \epsilon \sim N(0, I_k)$
- keep $z'_{S_t} = z^{(t)}_{S_t}$

We accept z' with MH probability using $\tilde{\pi}$. This constitutes a Metropolis-within-Gibbs kernel, since only the block S_t is stochastically updated while the complement is deterministic.

During burn-in, we minimize the reverse-KL: $\mathcal{L}(\phi) = E_{z \sim p_Z}[logp_Z(z) - log\tilde{\pi}_\phi(z)]$. After the burn-in, ϕ is frozen.

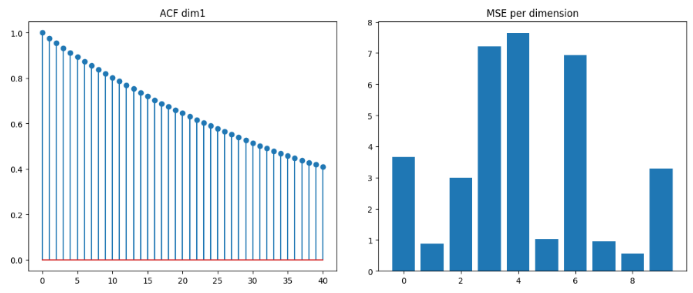

Figure 1: ACF and MSE for experiment 4.2 - NF MCMC

3.2.1 Differences between the work and GFlowMC

- Online vs. offline flow tuning.
- Simpler affine RealNVP instead of spline-coupling.
- Constant block fraction (10 %) vs. heuristic schedule.
- Lightweight accept-rate adaptation of σ; no second-order information required.

4 Experiments

4.1 Tricks to deal with high dimension

- Block updates: each MH step perturbs only 10 % of the coordinates selected at random
- Latent-space random walk + flow transport: the RealNVP maps an isotropic Gaussian step in z to a direction-aligned step in x. The NF learns global scaling and rotations, so the block move in z translates into a coherent, potentially large move in x even when the posterior covariance is ill-conditioned.
- Online flow adaptation during burn-in only
- Triangular Jacobian of RealNVP

4.2 3-component Gaussian mixture

The benchmark target is a 3-component Gaussian mixture in $\Re^{10}$ with well-separated means. Such multimodal geometry is challenging for plain random-walk MH, which gets trapped in a single mode. NF-guided proposals warp the modes closer in latent space, letting the chain jump freely. Futhermore, the ESS of our approach was better than the baseline, which kept low values per dimension, on the other hand this work had ESS mean higher 1000 close to the end of iteration (Fig 2 and 3). The MSE values per cordinates of this work is better, but not so good yet and we can note a decrease of autocorrelation in the dimensions, as we can see in the dimension 1 for example (Fig 1), so some changes have to be done, such as more iterations and test others hyperparameters of the NF.

In our code in GitHub, we tested for $\Re^{20}$ too and, as expected, the results aren't good as the previous one, but

4.3 UCI Adult Income

The UCI "Adult Income" dataset is a classic binary-classification benchmark, originally extracted from the 1994 U.S. Census database. Its goal is to predict whether an individual's annual income exceeds \$50 000 (the positive class) based on a mix of demographic and employment attributes. Below is a concise description of the problem setup. This dataset has 14 features and the target Binary label.

In the Bayesian logistic-regression experiment on the Adult Income data set, we first convert the eight categorical variables to one-hot indicators, i.e. each categorical variable is expanded into a separate binary column for every distinct level. The final dimension is, ending up with a $\approx$ 50-dimensional feature vector per record. The weight vector $w \in \Re^{50}$ is given by a prior Gaussian $N(0, \sigma I)$ and the likelihood is Bernoulli with logit Xw. To draw posterior samples we employ our NF-MCMC. Each MCMC iteration perturbs roughly 15 % of the coordinates of z via a random-walk step; the proposal is mapped to w by the flow and accepted or rejected with the Metropolis–Hastings rule computed in latent space. After burn-in the flow is frozen, ensuring a time-homogeneous Markov chain.

Although the results were not so good, more tests could be done and the use of Neural Spline Coupling to substitute the Affine. The best test accuracy was 88 % and as we can see, the ESS wasn't good enough and the space of possibilities wasn't so much explored (Fig 4).

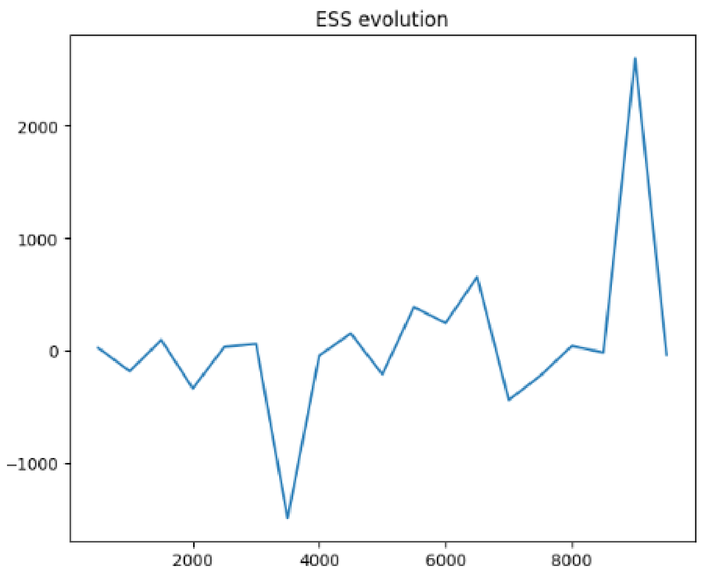

Figure 2: ESS mean of NF MCMC per iteration for experiment 4.2

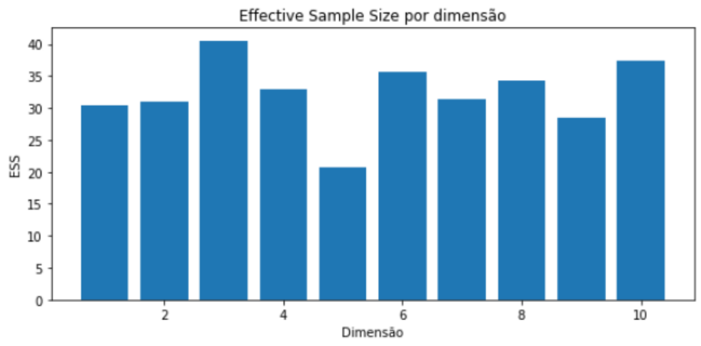

Figure 3: ESS per dimension for experiment 4.2 - baseline MCMC

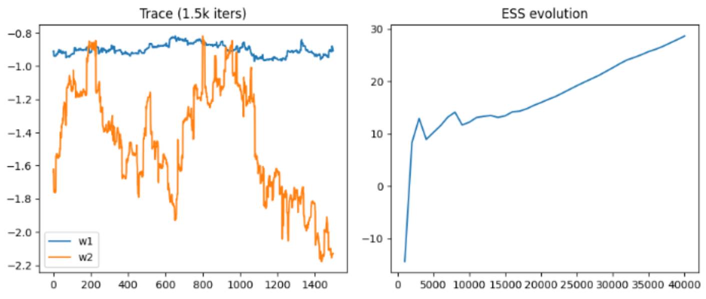

Figure 4: ESS and trace plot for experiment 4.3

5 Conclusion

Embedding a normalizing flow inside a block-wise MH sampler provides a simple yet powerful route to accelerate MCMC. On a ten-dimensional mixture the NF-assisted chain achieved ≈ 16 larger ESS and lower mean-squared-error than a tuned random-walk Metropolis baseline, while maintaining a lightweight implementation based on RealNVP and a single adaptive phase. The flow learns global re-scalings and non-linear couplings during burn-in, after which the Markov kernel remains time-homogeneous and theoretically sound. Although we focused on affine RealNVP couplings, preliminary tests with spline transformations suggest further gains are available with richer flows.

That said, not all outcomes were equally encouraging. In the logistic-regression experiment the predictive log-loss and the absolute ESS remain sub-optimal, indicating that the chain still struggles to explore regions of high posterior probability. Several factors likely contributed to these shortcomings, such as the limited flow capacity, because a ten-layer affine RealNVP may be unable to capture the highly skewed posterior geometry, especially for coefficients tied to rare categorical levels. In summary, while NF-MCMC offers an appealing compromise between ease of implementation and sampling efficiency, its performance depends on flow expressiveness and adaptive hyper-parameters. Future work should therefore, for example, to incorporate spline or autoregressive couplings to model heavy-tailed directions.

6 References

1. Papamakarios, G., Nalisnick, E., et al. (2021). Normalizing Flows for Probabilistic Modeling and Inference. JMLR.
2. Naesseth, C. A., Linderman, S. W., et al. (2018). Variational Sequential Monte Carlo. ICML.
3. Thomas B. Schon, Fredrik Lindsten, Johan Dahlin, Johan Wagberg, Christian A. Naesseth, Andreas Svensson and Liang Dai. Sequential Monte Carlo methods for system identification. In Proceedings of the 17th IFAC Symposium on System Identification (SYSID), Beijing, China, October 2015
4. A. Doucet, N. De Freitas, and N. Gordon. An introduction to sequential Monte Carlo methods. In Sequential Monte Carlo methods in practice. Springer-Verlag New York, 2001.
5. Schönle, C., & Gabrié, M. (2023, outubro 28). Optimizing Markov Chain Monte Carlo Convergence with Normalizing Flows and Gibbs Sampling. Poster apresentado no NeurIPS 2023 – AI4Science Workshop
6. Cabezas, A., Sharrock, L., & Nemeth, C. (2024, maio 23). Markovian Flow Matching: Accelerating MCMC with Continuous Normalizing Flows
7. Robert, C. P., & Casella, G. (2004). Monte Carlo statistical methods (2nd ed.). Springer.
8. Givens, G. H., & Hoeting, J. A. (2012). Computational statistics (2nd ed.). John Wiley & Sons.