Machine Learning for Rare Event Sampling

V

2025

1 Introduction

Standard Markov chain Monte Carlo (MCMC) [Itzykson and Drouffe, 1989] methods can be inefficient at sampling rare events such as phase transitions. Near criticality, simulations become increasingly expensive due to the local nature of typical update proposals, resulting in large autocorrelation times. Recent approaches [Albergo et al., 2019] leverage normalizing flows (generative models) to learn an invertible map from a simple prior (e.g., a Gaussian) to an approximation of the Boltzmann distribution of the physical system.

2 Discretization of the theory

In this section we derive the lattice discretization of the scalar ϕ^4 theory that will be used as target distribution for both standard MCMC and flow-based sampling. We start from a generic action in $1+1$ dimensions

$$S(\phi) = \int dt\, dx\, \left(\frac{1}{2}(\partial_\mu \phi)^2 - V(\phi)\right). \tag{1}$$

The quartic action then reads

$$S(\phi) = \int dt\, dx\, \left(\frac{1}{2}(\partial_\mu \phi)^2 - \frac{1}{2}m^2\phi^2 - \frac{\lambda}{4!}\phi^4\right). \tag{2}$$

2.1 Wick rotation

The first step is to perform a Wick rotation,

$$t \to -i\tau, \tag{3}$$

which maps real time to Euclidean time (an inverse temperature). In this way we can write the Euclidean action

$$S_E(\phi) = \int_0^\beta d\tau\, \int dx\, \left(\frac{1}{2}(\partial_\tau\phi\partial_\tau\phi + \partial_x\phi\partial_x\phi) + V(\phi)\right). \tag{4}$$

2.2 Integration by parts

Another useful trick is to integrate by parts to rewrite first-derivative terms in terms of second derivatives:

$$\int_\Omega (\partial\phi)^2 \to (\phi\,\partial\phi)_{\partial\Omega} - \int_\Omega \phi\,\partial^2\phi, \tag{5}$$

For the spatial case, we are considering an infinite volume Ω_S, that is it has no boundary $\partial\Omega_S = 0$, so we have

$$\int_{\Omega_S} dx\,(\partial_x\phi)^2 \to -\int_{\Omega_S} \phi\,\partial_x^2\phi. \tag{6}$$

For the temporal case, instead, we recall that the boundary conditions for the Euclidean action are

$$\phi(0) = \phi(\beta), \tag{7}$$

that is, in this case too, there are no terms due to the boundaries

$$\int_0^\beta d\tau\,(\partial_\tau \phi)^2 \to -\int_0^\beta d\tau\,\phi\,\partial_\tau^2 \phi. \tag{8}$$

2.3 Discretization

As the potential $V(\phi)$ depends only on the field, its discretization is quite easy, we just have to define the lattice.
Suppose we discretize the temporal axis with steps Δt, so that

$$t = n\Delta t, \quad n \in \mathbb{N}, \tag{9}$$

and we discretize the spatial axis with steps Δx, so that

$$x = j\Delta x, \quad j \in \mathbb{N}. \tag{10}$$

Then the discretization of the potential is just

$$V(\phi(x,\tau)) \to V(\phi_j^n), \tag{11}$$

where we use the subscript j to indicate the spatial index and the superscript n to indicate the temporal one.
The discretization of the kinetic term requires more attention due to the presence of the derivatives. The key idea is that, having used the integration by parts, the second derivative simplifies our calculations as it preserves the parity symmetry $x \to -x, \tau \to -\tau$, in fact we can write the discretized second derivative as

$$\partial_x^2 \phi \to \frac{\phi_{j+1} + \phi_{j-1} - 2\phi_j}{\Delta x^2} \tag{12}$$

and in the same way for the temporal case. We then have that

$$\phi \partial_x^2 \phi \to \phi_j \frac{\phi_{j+1} + \phi_{j-1} - 2\phi_j}{\Delta x^2}, \tag{13}$$

while the measure is discretized as follows

$$\int d\tau\,dx \to \sum_{j,n} \Delta t \Delta x. \tag{14}$$

Then putting all together, for example for the spatial case, we have

$$\int d\tau\,dx\,\frac{1}{2}\phi\,\partial_x^2\phi \to \sum_{j,n} \Delta t \Delta x \frac{1}{2}\phi_j \frac{\phi_{j+1} + \phi_{j-1} - 2\phi_j}{\Delta x^2}, \tag{15}$$

which we can rewrite, using the fact that we are summing over the j index, as

$$\int d\tau\,dx\,\frac{1}{2}\phi\,\partial_x^2\phi \to \sum_{j,n} \frac{\Delta t}{\Delta x}\frac{1}{2}(2\phi_{j+1}\phi_j - 2\phi_j^2). \tag{16}$$

We can do the same for the temporal case

$$\int d\tau\,dx\,\frac{1}{2}\phi\,\partial_\tau^2\phi \to \sum_{j,n} \frac{\Delta x}{\Delta t}\frac{1}{2}(2\phi^{n+1}\phi^n - 2(\phi^n)^2). \tag{17}$$

We have omitted the other indices for both case for simplicity. The final form for the discretized action will then be

$$S_E(\phi) = \sum_{j,n} \left[-\frac{\Delta x}{\Delta t}(\phi_j^{n+1}\phi_j^n - (\phi_j^n)^2) - \frac{\Delta t}{\Delta x}(\phi_{j+1}^n\phi_j^n - (\phi_j^n)^2) + \Delta t \Delta x V(\phi_j^n) \right] \tag{18}$$

2.4 Numerical case

In our numerical case we set $\Delta x = \Delta t = 1$ and we work with a quartic potential, so the action takes the form

$$S_E(\phi) = \sum_{j,n} \left[((\phi_j^n)^2 - \phi_j^{n+1}\phi_j^n) + ((\phi_j^n)^2 - \phi_{j+1}^n \phi_j^n) + \frac{m^2}{2}(\phi_j^n)^2 + \lambda(\phi_j^n)^4 \right] \tag{19}$$

which we can rewrite as

$$S_E(\phi) = \sum_{j,n} \left[-(\phi_j^{n+1}\phi_j^n + \phi_{j+1}^n\phi_j^n) + \frac{(m^2+4)}{2}(\phi_j^n)^2 + \lambda(\phi_j^n)^4 \right] \tag{20}$$

or as

$$S_E(\phi) = \sum_x \left[-\sum_\mu \phi_x \phi_{x+\mu} + \frac{m^2+4}{2}\phi_x^2 + \lambda\phi_x^4 \right] \tag{21}$$

3 Metropolis algorithm

In this section we present the Metropolis algorithm that we use to sample the target distribution of our system, to do this we first review briefly the concept of Markov chain which is the basis of MCMC methods. A Markov chain is a stochastic process in which the probability of transitioning to a new configuration depends only on the current configuration.
We are interested in sampling this probability distribution with Monte Carlo methods, in particular, we want to sample the final distribution which is independent from our starting point and so it is also called **equilibrium** distribution.

3.1 Metropolis algorithm

The Metropolis algorithm is our way to sample the equilibrium distribution starting from a generic initial state.
Let us consider a lattice, the algorithm is really simple to implement:

1. Select a site;
2. Trial state;
3. Accept/Reject.

We start by selecting a random state r in our lattice field ϕ; the second step is to define a trial variable/state depending on our model: for example in the field theory we define the trial state as

$$\hat{\phi}_r = \phi_r + \Delta(2u - 1), \tag{22}$$

where u is a random variable between 0 and 1 and Δ is a parameter of the metropolis algorithm that we can choose. Another case is the classical Ising model where we take as trial variable the flipped spin on that site

$$\hat{\phi}_r = -\phi_r. \tag{23}$$

Having defined the trial state $\hat{\phi}_r$ we accept it as the new state in our lattice with probability

$$\exp(-\delta S), \tag{24}$$

where δS is the difference between the action evaluated in the new state and the one evaluated in the old state, that is

$$\delta S = S[\hat{\phi}] - S[\phi]. \tag{25}$$

Accepting the state with such probability means extracting a random variable A between 0 and 1 and accepting the state if

$$A < \exp(-\delta S), \tag{26}$$

otherwise we reject the state. In other words the acceptance probability is

$$A(\phi \to \hat{\phi}) = \min(1, \exp(-\delta S)). \tag{27}$$

This choice ensures detailed balance with respect to the target distribution

$$p(\phi) \propto \exp(-S(\phi)). \tag{28}$$

Notice that if $\exp(-\delta S) \geq 1$ the move is always accepted.

4 Integrated Autocorrelation Time

To check if our Markov Chain has thermalized we can evaluate the integrated autocorrelation time τ of some observable, which is defined as follows

$$\tau = \frac{1}{2} + \sum_{k=1}^{W} \rho(k), \tag{29}$$

where $\rho(k)$ are the normalized autocorrelations

$$\rho(k) = \frac{\Gamma(k)}{\Gamma(0)}, \tag{30}$$

and

$$\Gamma(k) = \langle A(t)A(t+k) \rangle, \tag{31}$$

with $A(t)$ observable with zero mean. The parameter W, on the other hand, is just a way to evaluate the sum numerically, as it cannot be an infinite sum, and it is calculated with the Sokal method.
In our work $A(t)$ is the absolute magnetization of our field.

4.1 Some results

Let us now present some results for τ for different values of the parameters, where we have measured the absolute magnetization of our field.
We fix

$$L = 16,\ \lambda = 1.0,\ N_{\text{samples}} = 10^4,\ N_{\text{therm sweeps}} = 10^4 \tag{32}$$

and we vary the mass squared m^2

$$\begin{bmatrix} m^2 \\ \tau \end{bmatrix} = \begin{bmatrix} -4.5 & -4 & -3 \\ 34.21 & 161.20 & 213.26 \end{bmatrix} \tag{33}$$

The higher values of τ as m^2 increases mean that we are approaching the critical point and as so we need to increase our number of sweeps to truly catch the underlying dynamics (critical slowing down).
Below in Fig.1 we show some plots of the absolute magnetization as the markovian time varies.

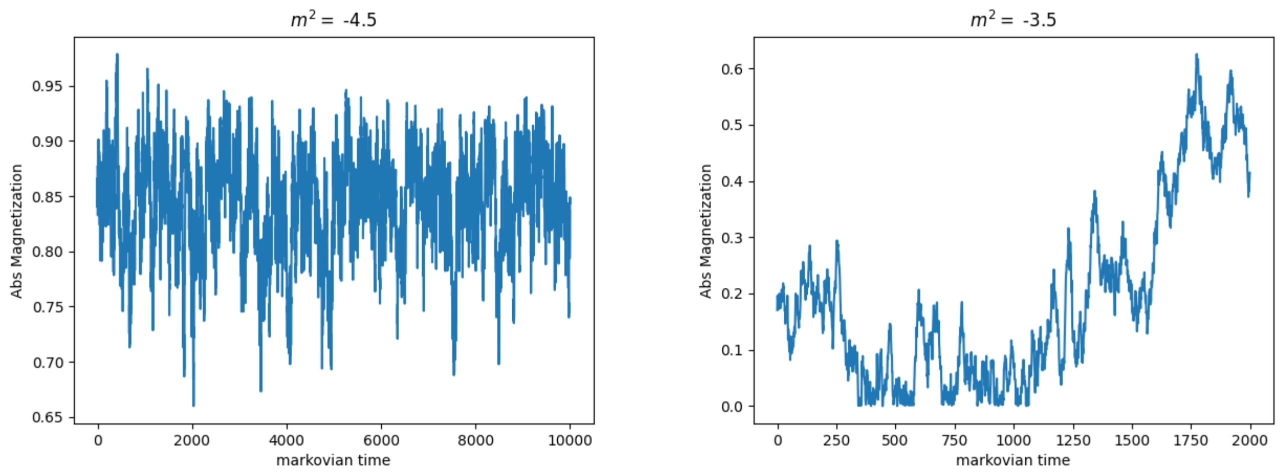

Figure 1: Absolute magnetization as a function of Markov-chain time for $m^2 = -4.5$ and $m^2 = -3.5$.

5 Flow based MCMC

To address critical slowing down, we follow the approach of [Albergo et al., 2019] based on normalizing flows: flow-based MCMC proposes a global configuration at each Markov-chain step, which can substantially reduce autocorrelations.
These flows learn a map f^{-1} (f is called *flow*) to evaluate the target distribution $\tilde{p}_f$ and use it in the Metropolis algorithm 3.1 to sample the true distribution of our system. The main advantage of this approach is the self-training, that is we do not need real data to train the model, but only the action of the system which enters in the (shifted) Kullback-Leibler (KL) divergence, that is the loss that we want to minimize.

5.1 Flows

The main idea of [Albergo et al., 2019] is enclosed in the diagram 2: where z are samples drawn from a

$$\begin{tikzcd} z \arrow[d] & \phi \arrow[l, "f"'] \arrow[d] \\ r \arrow[r] & \tilde{p}_f \end{tikzcd}$$

Figure 2: Diagram of the flow based MCMC.

simple (prior) distribution r: then f^{-1} is a change of variables allowing us to sample ϕ according to the new distribution $\tilde{p}_f$, which is close to the effective, physical, distribution

$$p(\phi) = \frac{e^{-S(\phi)}}{Z}. \tag{34}$$

Having said that, we construct the map f using the real non-volume-preserving (NVP) flow approach, that is we construct f by composition of affine coupling layers $\{g_i\}$

$$f(\phi) = g_1 \circ g_2 \circ \ldots g_n(\phi). \tag{35}$$

The proposal distribution $\tilde{p}_f$ is then given by

$$\tilde{p}_f(\phi) = r(f(\phi)) \left|\det \frac{\partial f(\phi)}{\partial \phi}\right|.$$

To retrieve the target distribution $p(\phi)$ we use the Metropolis algorithm 3.1 with acceptance probability

$$A(\phi \to \phi') = \min\left(1, \frac{p(\phi')\,\tilde{p}_f(\phi)}{p(\phi)\,\tilde{p}_f(\phi')}\right). \tag{36}$$

5.2 Affine coupling layers

Given a field configuration ϕ, affine coupling layers g_i return a masked field ϕ'.
We start by dividing the field ϕ into two halves ϕ_a, ϕ_b, then the affine coupling g_i acts as follows:

$$\begin{cases} \phi_a \to \phi_a & \text{(one half is invariant)} \\ \phi_b \to z_b = \phi_b \odot e^{s_i(\phi_a)} + t_i(\phi_a) & , \end{cases} \tag{37}$$

where $\odot$ is the element-wise multiplication and s_i, t_i are (learned) functions parametrizing the affine coupling.
After every layer we swap ϕ_a and ϕ_b, i.e., we alternate which half is transformed.
We can now easily compute the jacobian recalling that the final jacobian is just the product of the single jacobian, so our task is to compute

$$\left|\frac{\partial g_i(\phi)}{\partial \phi}\right|. \tag{38}$$

We first notice that half of the coupling is just the identity ($\phi_a \to \phi_a$), so we just need to derive the other mapping which, because we are performing an element wise multiplication is simply

$$\frac{\partial g_i}{\partial(\phi_b)_j} = e^{(s_i)_j}. \tag{39}$$

Putting all together, if ϕ can be rearranged in a D-dimensional vector, we have

$$\left|\frac{\partial g_i(\phi)}{\partial \phi}\right| = \prod_{j=1}^{D/2} e^{(s_i(\phi))_j}. \tag{40}$$

5.3 Loss function

It is important now to discuss the loss function that we want to minimize to train our flow: the shifted Kullback-Leibler (KL) divergence between the target distribution $p(\phi)$ and the proposal distribution $\tilde{p}_f(\phi)$.
The KL divergence is defined as

$$L(\tilde{p}_f) = D_{KL}(\tilde{p}_f||p) - \log Z \tag{41}$$

and contains the logarithm of the partition function simplifying the calculations. D_{KL} is defined as

$$D_{KL}(\tilde{p}_f||p) = \int d\phi\,\tilde{p}_f(\phi)\log\left(\frac{\tilde{p}_f(\phi)}{p(\phi)}\right), \tag{42}$$

so we can rewrite the loss as

$$L(\tilde{p}_f) = \int d\phi\,\tilde{p}_f(\phi)\left[S(\phi) + \log\tilde{p}_f(\phi)\right]. \tag{43}$$

It is important to notice that the KL-divergence D_{KL} is always non-negative and it is zero if and only if the two distributions are equal almost everywhere, so minimizing the loss means making the two distributions as similar as possible.

5.4 Implementation details

Let us now discuss how to implement the flow based MCMC, which is done in python using the Jax and the Flax frameworks to make use of GPUs. The core lies in the choice of the model, specifically we consider a Convolutional Neural Network (CNN) which is really suited for lattice field theories as it can capture the spatial correlations between nearby sites.
The CNN is built using $N_{\text{layers}} = 32$ affine coupling layers, each of them made by two convolutional layers with kernel size 3×3 with ReLU activations. We have also implemented PBC for the CNN to respect the boundary conditions of the lattice, while the training is performed using the AdamW optimizer.

5.5 Results for flow based MCMC

Before discussing the results for the flow based MCMC, a consideration is in order: in the flow based MCMC every configuration is a global configuration, so we do not need to perform any sweeps between two samples. Fixing the parameters as the ones in the metropolis case for comparison, that is

$$L = 16, \quad m^2 = -3.0, \quad \lambda = 1.0, \quad N = 5000, \tag{44}$$

we get as autocorrelation time $\tau = 19.69$ with an acceptance rate of 17%. In Fig. 3 we show the absolute magnetization as a function of the markovian time. In Fig. 4, instead, we show how the normalized auto-

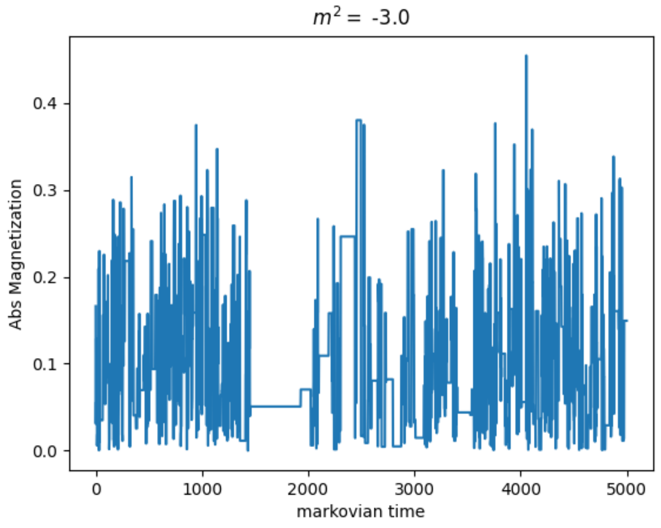

Figure 3: Absolute magnetization for a configuration sampled with the flow based MCMC.

correlations $\rho(k)$ (30) behave as a function of the lag k in both the flow based case and the simple MC case.

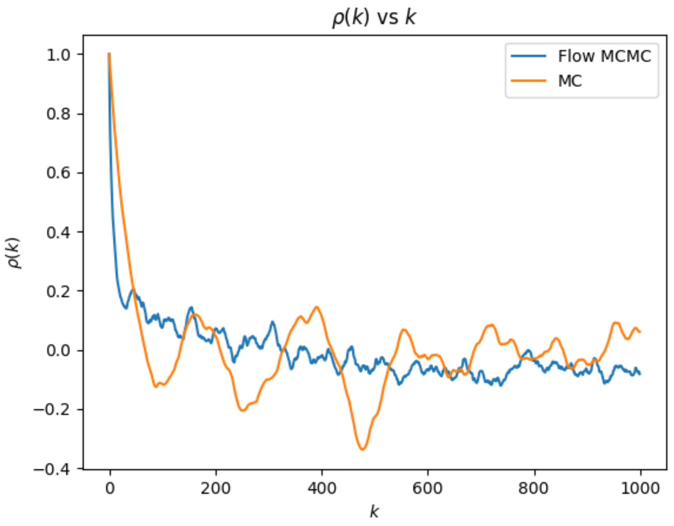

Figure 4: Comparison of the normalized autocorrelation function $\rho(k)$ for Standard MCMC (orange) and Flow-based MCMC (blue). The Flow-based method decorrelates significantly faster.

6 Conclusion

In this work we have presented the lattice discretization of the scalar ϕ^4 theory in $1+1$ dimensions and we have implemented both a standard Metropolis MCMC and a flow based MCMC to sample the target distribution of the system.
We have shown that the standard MCMC suffers from critical slowing down near the critical point, while the flow based MCMC is able to sample the distribution with a really low autocorrelation time thanks to the global nature of the generated configurations.
It is important now to show the clear limitations of this approach: lattice sizes higher than $L = 16$ are really challenging to simulate as the training of the flow becomes really unstable. This point will be the focus of future work.

References

[Albergo et al., 2019] Albergo, M. S., Kanwar, G., and Shanahan, P. E. (2019). Flow-based generative models for markov chain monte carlo in lattice field theory. *Physical Review D*, 100(3).

[Itzykson and Drouffe, 1989] Itzykson, C. and Drouffe, J. M. (1989). *STATISTICAL FIELD THEORY. VOL. 1: FROM BROWNIAN MOTION TO RENORMALIZATION AND LATTICE GAUGE THEORY.* Cambridge Monographs on Mathematical Physics. CUP.