Self-learning Monte Carlo method

Junwei Liu,[1,*] Yang Qi,[1] Zi Yang Meng,[2] and Liang Fu[1,†]

[1]*Department of Physics, Massachusetts Institute of Technology, Cambridge, Massachusetts 02139, USA*
[2]*Institute of Physics, Chinese Academy of Sciences, Beijing 100190, China*

Monte Carlo simulation is an unbiased numerical tool for studying classical and quantum many-body systems. One of its bottlenecks is the lack of a general and efficient update algorithm for large size systems close to the phase transition, for which local updates perform badly. In this Rapid Communication, we propose a general-purpose Monte Carlo method, dubbed self-learning Monte Carlo (SLMC), in which an efficient update algorithm is first learned from the training data generated in trial simulations and then used to speed up the actual simulation. We demonstrate the efficiency of SLMC in a spin model at the phase transition point, achieving a 10–20 times speedup.

The Monte Carlo (MC) method is a powerful and unbiased numerical tool for simulating statistical and condensed matter systems [1–5]. MC simulation obtains statistically exact values of physical observables by sampling a large number of configurations according to the Boltzmann distribution. Configurations can be generated sequentially by the local update method [6,7]. However, when the system is close to a phase transition, the local update can be highly inefficient as sequentially generated configurations are strongly correlated, causing a significant slowing down in the simulation dynamics. For certain classes of models, this slowing down can be overcome by global update methods [8–14], where an extensive number of local variables are changed in a single update. However, for any generic model, it is highly challenging to design an efficient global update method.

Inspired by great developments in machine learning [15], in this Rapid Communication we propose an approach to speed up the MC simulation. The MC sampling process generates a sequence of configurations in a Markov chain, which constitutes a massive set of data containing valuable information about the system. Meanwhile, machine learning is a powerful technique to uncover unknown properties in the data and make new predictions. Thus we expect that machine learning can extract the information hidden in the Markov chain, which we then use to improve the performance of MC simulation.

Specifically, we propose an MC update method applicable to generic statistical models, dubbed the self-learning Monte Carlo (SLMC) method. The essence of SLMC is to first perform a trial simulation with a local update to obtain a sequence of configurations and their weights, serving as the training data, and then to learn a rule that guides the configuration update in an actual simulation. To demonstrate the power of SLMC, we study a statistical model [see Eq. (1)] for which no efficient global update scheme is known. We find that, in comparison to the local update, SLMC significantly reduces the autocorrelation time, especially near the phase transition.

Outline of SLMC. Before presenting our method, let us recall that configurations in MC simulation can be updated through a Markov process, where the transition probability from configuration A to B, $P(A \rightarrow B)$, is required to satisfy the detailed balance principle (DBP) [6], $P(A \rightarrow B)/P(B \rightarrow A) = W(B)/W(A)$, where W is the probability distribution of configurations. Update methods can be roughly divided into two types: local and global.

The local update is a general-purpose, model-independent method, consisting of two steps. First, one randomly chooses a single site in the current configuration and proposes a new configuration by changing the variable on this site. Second, one decides whether the proposed move is accepted or rejected based on DBP. If accepted, the next configuration in the Markov chain will be the new one; otherwise, it will be a copy of the current one. Clearly, the way a local move is proposed in the first step is completely general and does not use any knowledge of the model. The local update works well for many systems, but suffers heavily from a critical slowing down close to phase transitions [8,9]. In such cases, the autocorrelation time within the Markov chain τ becomes very large, and in fact diverges with the system size L as $\tau \sim \tau_0 L^z$ at the critical points, where z is the dynamical exponent of MC simulation.

To overcome the dramatic increase of autocorrelation time for the local update, many global update methods have been developed, such as Swendsen-Wang [8], Wolff [9], worm [10], loop [11,12], and directed loop [13,14] algorithms. In all these methods, variables on an extensive number of sites are simultaneously changed in a single MC update, thus reducing the dynamic exponent z significantly. However, unlike the local update, here the proposal of a trial configuration and the determination of its acceptance are intricately linked, because the proposed move already takes into account the DBP. Thus global updates are ingeniously designed methods targeted for special models. For a given generic model, it is very difficult to design an efficient global update method.

From a comparison of local and global updates, we conclude that a general-purpose MC update method that can outperform the local update must satisfy the following requirements: (1) A large number of sites should be involved in each move that updates the current configuration; (2) the proposal and the acceptance of moves should be independent. For systems at the critical point, we further require the number

―――――――
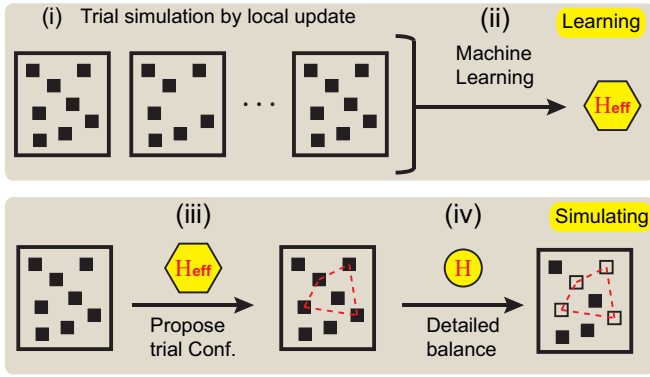*liujw@mit.edu
†liangfu@mit.edu

FIG. 1. Schematic illustration of the learning process (top panel) and simulating process (bottom panel) in self-learning Monte Carlo.

of sites involved in each move to increase with the system size in order to reduce the dynamical exponent z in MC simulation.

Guided by these requirements, we now propose the detailed procedure of the SLMC method. As shown in Fig. 1, SLMC consists of four steps: (i) Perform a trial MC simulation using the local update to generate a large number of configurations, which serve as the training data; (ii) learn an effective Hamiltonian H_{eff} from this training data; (iii) propose moves according to H_{eff} in the actual MC simulation; and (iv) determine whether the proposed moves will be accepted or rejected based on the detailed balance principle of the *original* Hamiltonian H. Steps (i) and (ii) constitute the learning process, whereas steps (iii) and (iv) are repeated in the actual MC simulation to calculate physical observables.

We further outline how to implement steps (ii) and (iii) in actual simulations for a model to be presented below. We use machine learning [15] in step (ii) to train an effective Hamiltonian, which can be efficiently simulated using a global update method even though the original Hamiltonian cannot. Then step (iii) can be easily implemented using this global update.

Model and results. To demonstrate the power of SLMC, we study a classical model on a two-dimensional (2D) square lattice,

$$H = -J \sum_{\langle ij \rangle} S_i S_j - K \sum_{ijkl \in \square} S_i S_j S_k S_l, \tag{1}$$

where $S_i = \pm 1$ is the Ising spin on site i. J is the nearest-neighbor (NN) interaction and K is the interaction among the four spins in the same plaquette. We set ferromagnetic interactions, i.e., $J > 0$ and $K > 0$. For any finite J and K, there is a phase transition from the paramagnetic phase at high temperature to the ferromagnetic phase at low temperature, which belongs to the 2D Ising universality class. For $K = 0$, this model reduces to the standard Ising model which can be simulated efficiently by the Wolff method. However, for $K \neq 0$, no simple and efficient global update method is known. Below we will show that the SLMC method significantly reduces the autocorrelation time near the critical point, using $K/J = 0.2$ as an example. More results can be found in the Supplemental Material (SM) [16].

As outlined before, the initial step of the SLMC is to train an effective Hamiltonian H_{eff} from a sample of

TABLE I. The trained parameters $\{\tilde{J}_n\}$ of the effective model in Eq. (2), without and with setting $\tilde{J}_n = 0$ ($n \geqslant 2$).

	$\tilde{J}_1$	$\tilde{J}_2$	$\tilde{J}_3$	Mean error
Train 1	1.2444	-0.0873	-0.0120	0.0009
Train 2	1.1064			0.0011

configurations generated by the local update based on the original Hamiltonian in Eq. (1). We choose H_{eff} to be a generalized Ising Hamiltonian with two-body spin interactions over various ranges,

$$H_{\text{eff}} = E_0 - \tilde{J}_1 \sum_{\langle ij \rangle_1} S_i S_j - \tilde{J}_2 \sum_{\langle ij \rangle_2} S_i S_j - \cdots, \tag{2}$$

where $\langle ij \rangle_n$ denotes the nth NN interaction and $\tilde{J}_n$ is the corresponding interaction parameter.

We now train H_{eff} from the training sample by optimizing E_0 and $\{\tilde{J}_n\}$. In principle, this can be viewed as an unsupervised learning process [15,17], where a new statistical model H_{eff} is trained using a subset of features extracted from the configurations. However, by taking advantage of knowing H for each configuration, we can more efficiently train H_{eff} through a simple and reliable linear regression. For the ath configuration in the sample, we compute its energy E^a [from Eq. (1)] and all the nth NN spin-spin correlations $C_n^a = \sum_{\langle ij \rangle_n} S_i S_j$, which serve as the actual training data. Then, E_0 and $\{\tilde{J}_n\}$ can be easily trained from a multilinear regression of E^a and $\{C_n^a\}$, $E^a = \sum_n \tilde{J}_n C_n^a + E_0$. The results are as shown in Table I (Train 1). It is clear that $\tilde{J}_1$ is dominant and much larger than others, which implies we could set $\tilde{J}_n = 0$ ($n \geqslant 2$). And then, by a linear regression, we can successfully extract the most optimized $\tilde{J}_1$ (Train 2 in Table I). It is found that the mean error is almost the same as the case without setting $\tilde{J}_n = 0$ ($n \geqslant 2$), which is expected since all $\tilde{J}_n$ ($n \geqslant 2$) obtained from the multilinear regression are negligible. Through this training process, we conclude that only the nearest interaction is relevant there, thus we only keep this term in the following simulations. We emphasize that this trained model H_{eff} only approximates the original one for the configurations that are statistically significant in the sample, i.e., the ones near the free-energy minimum. Thus H_{eff} can be regarded as an effective model. We notice that, recently, there are many other attempts to apply machine learning to MC simulations [18–23].

In addition, it should be addressed that the training of H_{eff} could be self-improved by a reinforced learning process. Usually, a good initial sample could be very hard to generate using only the local update, especially for systems at the critical temperature T_c or with strong fluctuation. In this case, we first train an effective model H_{eff} using a simulation at temperature $T > T_c$, and then generate another sample at T_c, using the self-learning update with H_{eff} learned from the first iteration. Later, a more accurate H_{eff} can be learned from the second-iteration sample. In actual simulations, one can further improve this process by using more iterations, each done with a smaller sample. More details can be found in the Supplemental Material.
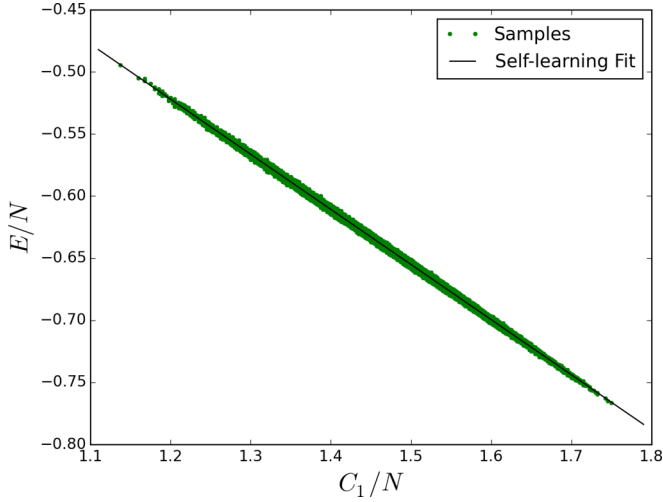
FIG. 2. Fitting of the distribution drawn from a sample of configurations in a Markov chain. The green dots represent configurations in the sample, for which the x axis shows the feature of the nearest-neighbor spin-spin correlation C_1, and the y axis shows the energy (per site) E/N computed from the original model in Eq. (1).

Through this iterative training process, we successfully arrive at the final H_{eff}. As shown in Fig. 2, H_{eff} (self-learning fit) indeed fits the energy of the configurations that are statistically significant in the simulation. In the main part of the figure, the data points are concentrated in the vicinity of the fitted line, indicating that trained H_{eff} is indeed a good description of the low-energy physics.

Following the procedure of SLMC, once the training process is finished, a cluster update with the Wolff algorithm according to H_{eff} can be constructed. Then, the generated cluster update is accepted or rejected with a probability accounting for the energy difference between the effective model and the original model. The probability of accepting a cluster is as follows,

$$\alpha(A \to B) = \min\{1, e^{-\beta[(E_B - E_B^{\text{eff}}) - (E_A - E_A^{\text{eff}})]}\}, \quad (3)$$

where A and B denote the configurations before and after flipping the cluster. E_A and E_A^{eff} denote the energies of a configuration A, for the original model in Eq. (1) and the effective model in Eq. (2), respectively. The derivation of Eq. (3) can be found in the SM. With Eq. (3), the detailed balance is satisfied, and the SLMC is *exact*, despite the use of an approximate effective model in constructing the cluster.

To test the efficiency of the update scheme in SLMC, we measure the autocorrelation time τ, which signifies how correlated the MC configurations are in the Markov chain (the detailed relation of τ with the computational complexity of MC algorithm can be found in SM). In Fig. 3, we plot τ of the ferromagnetic order parameter $M = \frac{1}{N}|\sum_i S_i|$, where N is the number of sites, measured at each step of the Markov chain, generated by different update algorithms on a square lattice of linear size $L = 40$. The simulation is done at T_c, which is determined by the Binder ratio, as shown in SM.

We compare results of the local update, the self-learning update using H_{eff}, and also a naive Wolff-like cluster update with the bare two-body J term from the original model in
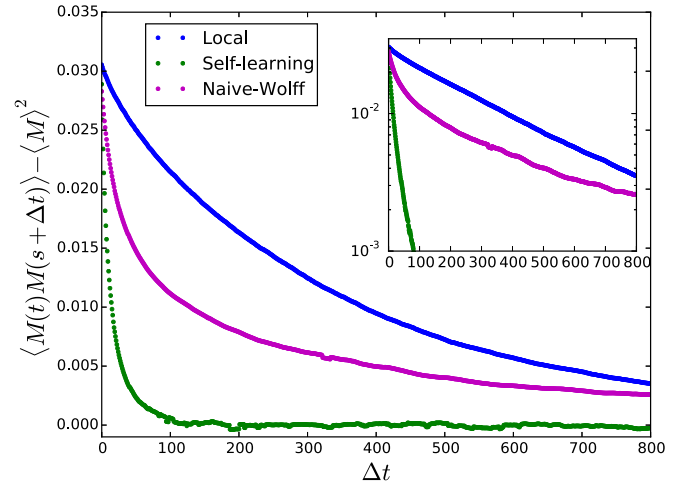

FIG. 3. The decay of autocorrelation functions as a function of MC steps, obtained using different update algorithms. Inset: Semilog plot of the same data.

Eq. (1) that is used to construct a cluster. The autocorrelation functions generated by all updates decay with the MC steps Δt, and the autocorrelation time τ can be obtained from fitting in the form of $e^{-\Delta t/\tau}$. Our results show that comparing to the local and naive cluster updates, the self-learning update has the much shorter τ. In particular, at this system size, the self-learning update is about 24 times faster than the local update, while the naive Wolff-like cluster update does not gain much of a speedup.

While Fig. 3 is an example of the better performance of SLMC for a fixed system size at T_c, we have further collected the autocorrelation time τ at T_c for local and self-learning updates with many different system sizes, and hence extract the scaling behavior of τ with respect to L. The results are shown in Fig. 4. The blue squares are the τ_L, i.e., autocorrelation time for local update, and it follows $\tau_L \sim L^{2.2}$, well consistent with the literature on critical slowing down [8,9]. The green dots are
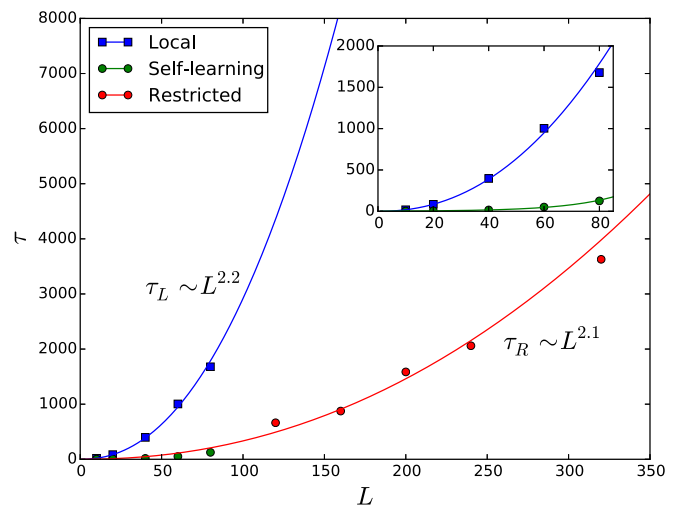

FIG. 4. The scaling behavior of the autocorrelation times of the local update τ_L, SLMC update τ_S, and the restricted SLMC update τ_R. The inset is a zoom-in for $L < 80$.

the τ_S, i.e., autocorrelation time for the self-learning update. For all the tested systems size $L \leqslant 80$, the τ_S delivers a large speedup of about 20 times (see the inset of Fig. 4 for clarity).

For a very large system size, we find τ_S increases exponentially with L, $\tau_S \propto e^{L/L_0}$ (more details in SM). This is because of a finite energy difference between the effective model in Eq. (2) and the original model in Eq. (1). Therefore, the acceptance ratio of flipping the whole cluster in Eq. (3) decreases exponentially as the length of the cluster boundary grows with increasing L, which renders the exponential increase of the autocorrelation time. But this drawback in SLMC can be easily remedied by simply restricting the maximum size of the cluster in the Wolff algorithm [24]. With this improvement, the averaged acceptance ratio can be expected to be fixed and SLMC should have the same scaling function for the autocorrelation time as the local update, $\tau_R = \tau_0 L^z$. However, by tuning the maximum size of the cluster, we can achieve a much smaller prefactor τ_0, and the optimized maximum cluster size can be automatically self-learned via a model-independent procedure (more details in SM). This is indeed the case. As shown by the red dots in Fig. 4, when the growth of the cluster is restricted to an area within 40 lattice spacing, the autocorrelation time τ_R becomes $\tau_R \propto L^{2.1}$, which obeys the same power law as τ_L, but with a prefactor of about 10 times smaller (more details about the design of this restricted SLMC are provided in SM). Therefore, although SLMC still suffers from a critical slowing down in the thermodynamic limit, we can gain a tenfold speedup. That means SLMC can achieve a much larger system size than the local update, which helps to overcome the finite-size effect. Moreover, for medium-sized systems, the SLMC without restriction can easily gain a 20-fold speedup, as shown by τ_S.

Discussion. We now discuss the applicability of the SLMC method to a broader class of problems in statistical and condensed matter systems. Besides spin systems, many models of great interest may be transformed into spin models with short-range interactions [5,25], for which efficient global update methods are available. In such cases, SLMC can be readily implemented similar to our model studied above. In particular, we expect SLMC to be very useful for studying strongly correlated fermion systems [26,27], where no efficient global update method is currently known. Moreover, by employing rapidly developing machine learning techniques, the SLMC method may be able to learn the configuration update on its own, without relying on a given form of the effective Hamiltonian. If realized, this will further increase the efficiency and versatility of SLMC.

SLMC may also bridge numerical and theoretical studies. The effective Hamiltonian trained or learned from the MC simulation may guide the theoretical study of the original model. The benefit is mutual: A theoretical understanding may improve the accuracy of the effective model and thus the performance of the numerical simulation.

Note added. Recently, a related work is described [28].

---

[1] *The Monte Carlo Method in Condensed Matter Physics*, edited by K. Binder (Springer, Berlin, 1995).

[2] M. E. J. Newman and G. T. Barkema, *Monte Carlo Methods in Statistical Physics* (Oxford University Press, Oxford, UK, 1999).

[3] *The Monte Carlo Method in the Physical Sciences: Celebrating the 50th Anniversary of the Metropolis Algorithm*, edited by J. E. Gubernatis, AIP Conf. Proc. Vol. 690 (AIP, Melville, NY, 2003).

[4] D. Landau and K. Binder, *A Guide to Monte Carlo Simulations in Statistical Physics* (Cambridge University Press, Cambridge, UK, 2005).

[5] A. W. Sandvik, in *Lectures on the Physics of Strongly Correlated Systems XIV: Fourteenth Training Course in the Physics of Strongly Correlated Systems*, edited by A. Avella and F. Mancini, AIP Conf. Proc. Vol. 1297 (AIP, Melville, NY, 2010), p. 135.

[6] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller, J. Chem. Phys. **21**, 1087 (1953).

[7] W. K. Hastings, Biometrika **57**, 97 (1970).

[8] R. H. Swendsen and J.-S. Wang, Phys. Rev. Lett. **58**, 86 (1987).

[9] U. Wolff, Phys. Rev. Lett. **62**, 361 (1989).

[10] N. Prokof'ev, B. Svistunov, and I. Tupitsyn, Phys. Lett. A **238**, 253 (1998).

[11] H. G. Evertz, G. Lana, and M. Marcu, Phys. Rev. Lett. **70**, 875 (1993).

[12] H. G. Evertz, Adv. Phys. **52**, 1 (2003).

[13] O. F. Syljuåsen and A. W. Sandvik, Phys. Rev. E **66**, 046701 (2002).

[14] F. Alet, S. Wessel, and M. Troyer, Phys. Rev. E **71**, 036706 (2005).

[15] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning* (Springer, New York, 2009).

[16] See Supplemental Material at http://link.aps.org/supplemental/10.1103/PhysRevB.95.041101 for more details and results.

[17] G. Torlai and R. G. Melko, Phys. Rev. B **94**, 165134 (2016).

[18] J. Carrasquilla and R. G. Melko, arXiv:1605.01735.

[19] L. Wang, Phys. Rev. B **94**, 195105 (2016).

[20] G. Carleo and M. Troyer, arXiv:1606.02318.

[21] P. Broecker, J. Carrasquilla, R. G. Melko, and S. Trebst, arXiv:1608.07848.

[22] K. Ch'ng, J. Carrasquilla, R. G. Melko, and E. Khatami, arXiv:1609.02552.

[23] H. J. Changlani, H. Zheng, and L. K. Wagner, J. Chem. Phys. **143**, 102814 (2015).

[24] G. T. Barkema and J. F. Marko, Phys. Rev. Lett. **71**, 2070 (1993).

[25] F. Assaad and H. Evertz, in *Computational Many-Particle Physics*, edited by H. Fehske, R. Schneider, and A. Weiße,

Lecture Notes in Physics Vol. 739 (Springer, Berlin, 2008), p. 277.

[26] J. Liu, H. Shen, Y. Qi, Z. Y. Meng, and L. Fu, arXiv:1611.09364.

[27] X.-Y. Xu, Y. Qi, J. Liu, L. Fu, and Z.-Y. Meng, arXiv:1612.03804.

[28] L. Huang and L. Wang, Phys. Rev. B **95**, 035105 (2017).