Guiding Monte Carlo Simulations with Machine Learning

Yang Qi

Department of Physics, Massachusetts Institute of Technology
Joining Fudan University in 2017

KITS, June 29 2017.

References

- Works led by Liang Fu, Zi-Yang Meng, and Lei Wang.
- Jun-Wei Liu, Xiao-Yan Xu, Hui-Tao Shen, Yuri Nagai, Li Huang, Yi-Feng Yang etc.
- References:

1. J Liu, YQ, Z-Y Meng and L Fu, PRB 95, 041101(R) (2017).
2. J Liu, H Shen, YQ, Z-Y Meng and L Fu, PRB 95, 241104(R) (2017).
3. X-Y Xu, YQ, J Liu, L Fu and Z-Y Meng, arXiv:1612.03804 [cond-mat.str-el].
4. Y Nagai, H Shen, YQ and L Fu, arXiv:1705.06724 [cond-mat.str-el].
5. Li Huang, Lei Wang, PRB 95, 035105(R) (2017).
6. Li Huang, Yi-Feng Yang, Lei Wang, PRE 95, 031301(R)(2017).
7. Lei Wang, arXiv:1702.08586 [cond-mat.str-el].

Outline

Outline

Monte Carlo simulation: an unbiased method

- A widely used numerical method in statistical physics and quantum many-body physics.
- Unbiased: reliable statistical error bar.
- Fast: error decreases as $1/\sqrt{\mathcal{N}}$.
- Universal: applies to any model without the sign program.

Introduction to MCMC

- Consider a statistical mechanics model:

$$Z = \sum_{\mathcal{C}} e^{-\beta H[\mathcal{C}]} = \sum_{\mathcal{C}} W(\mathcal{C}).$$

- The Markov-chain Monte Carlo (MCMC) is a way to do importance sampling.
- A Markov chain is constructed,

$$\cdots \to \mathcal{C}_{i-1} \to \mathcal{C}_i \to \mathcal{C}_{i+1} \to \cdots$$

- Markov chain: $p(\mathcal{C}_i \to \mathcal{C}_j)$ only depends on $\mathcal{C}_i$ (no memory).
- Goal: distribution of $\mathcal{C}$ converges to the Boltzmann distribution $W(\mathcal{C})$.
- Any observable can be measured from a Markov chain,

$$\langle O \rangle = \frac{\sum O(\mathcal{C}) W(\mathcal{C})}{\sum W(\mathcal{C})} \simeq \frac{1}{\mathcal{N}} \sum_i O(\mathcal{C}_i).$$

Detailed balance

- Detailed balance:
$$\frac{p(\mathcal{C} \to \mathcal{D})}{p(\mathcal{D} \to \mathcal{C})} = \frac{W(\mathcal{D})}{W(\mathcal{C})}.$$
- Detailed balance (and ergodicity) guarantees that IF the MC converges, it converges to the desired distribution $W(\mathcal{C})$.
- Metropolis-Hastings algorithm: propose – accept/reject.

$$p(\mathcal{C} \to \mathcal{D}) = q(\mathcal{C} \to \mathcal{D})\alpha(\mathcal{C} \to \mathcal{D}).$$

$$\alpha(\mathcal{C} \to \mathcal{D}) = \min\left\{1, \frac{W(\mathcal{D})}{W(\mathcal{C})}\frac{q(\mathcal{D} \to \mathcal{C})}{q(\mathcal{C} \to \mathcal{D})}\right\}.$$

Autocorrelation time

- Autocorrelation time measures the efficiency of the update algorithm.
- "time" sequence:

$$\cdots \to O(t-1) \to O(t) \to O(t+1) \to \cdots, \quad O(t) = O[\mathcal{C}(t)].$$

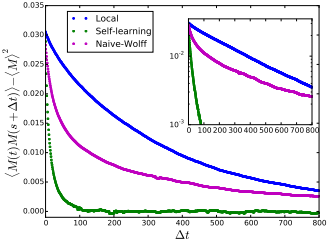
- Autocorrelation function

$$\mathcal{A}_O(\Delta t) = \langle O(t)O(t+\Delta t)\rangle - \langle O(t)\rangle^2 \propto e^{-\Delta t/\tau}.$$

Complexity $\propto \tau$

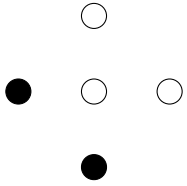
$$\langle O\rangle = \frac{1}{\mathcal{N}}\sum_i O(\mathcal{C}_i).$$

Statistical error $\delta O \sim \frac{1}{\sqrt{\mathcal{N}}}$ only if $O(i)$ are independent.



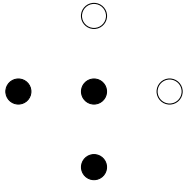
Outline

Metropolis algorithm



- Local update: randomly select a site and flip the spin.
- $q(\mathcal{C} \to \mathcal{D}) = q(\mathcal{D} \to \mathcal{C}) = \frac{1}{N}$.
- $\alpha(\mathcal{C} \to \mathcal{D}) = \min\left\{1, \frac{W(\mathcal{D})}{W(\mathcal{C})}\right\}$.
- N trials are counted as one MC step.
- Very general: applies to any model.
- N Metropolis, A W Rosenbluth, M N Rosenbluth, A H Teller, and E Teller, J Chem Phys **21**, 1087 (1953).

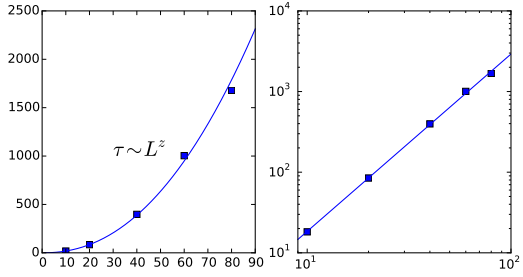
Metropolis algorithm



- Local update: randomly select a site and flip the spin.
- $q(\mathcal{C} \to \mathcal{D}) = q(\mathcal{D} \to \mathcal{C}) = \frac{1}{N}$.
- $\alpha(\mathcal{C} \to \mathcal{D}) = \min\left\{1, \frac{W(\mathcal{D})}{W(\mathcal{C})}\right\}$.
- N trials are counted as one MC step.
- Very general: applies to any model.
- N Metropolis, A W Rosenbluth, M N Rosenbluth, A H Teller, and E Teller, J Chem Phys **21**, 1087 (1953).

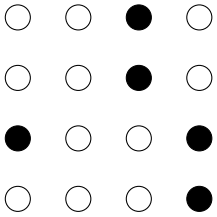
Critical slowing down

- The real dynamical relaxation time diverges at the critical point: a critical system is slow to equilibrate.
- The local update mimics the real relaxation process: also exhibits the critical slowing down phenomena.
- $\tau \propto L^z$, $z = 2.125$ for the 2D Ising model.



There's a way around this: MCMC simulation does not have to mimic the real dynamics...

Wolff algorithm: a cluster update

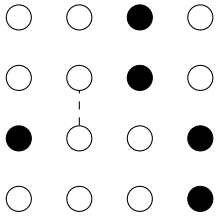


- A cluster is built one bond at a time.
- Probability of activating a bond is cleverly designed, such that

$$\frac{q(\mathcal{C} \to \mathcal{D})}{q(\mathcal{D} \to \mathcal{C})} = \frac{W(\mathcal{D})}{W(\mathcal{C})}.$$

- Thus we have the ideal acceptance ratio $\alpha = 1$.
- Very efficient update: $\tau \sim L^{0.35}$.
- Only works with two-body interactions: $H = -\sum_{ij} J_{ij} s_i s_j$.
- U Wolff, PRL. **62**, 361 (1989).

Wolff algorithm: a cluster update

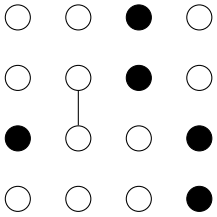


- A cluster is built one bond at a time.
- Probability of activating a bond is cleverly designed, such that

$$\frac{q(\mathcal{C} \to \mathcal{D})}{q(\mathcal{D} \to \mathcal{C})} = \frac{W(\mathcal{D})}{W(\mathcal{C})}.$$

- Thus we have the ideal acceptance ratio $\alpha = 1$.
- Very efficient update: $\tau \sim L^{0.35}$.
- Only works with two-body interactions: $H = -\sum_{ij} J_{ij} s_i s_j$.
- U Wolff, PRL. **62**, 361 (1989).

Wolff algorithm: a cluster update

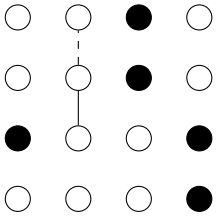


- A cluster is built one bond at a time.
- Probability of activating a bond is cleverly designed, such that

$$\frac{q(\mathcal{C} \to \mathcal{D})}{q(\mathcal{D} \to \mathcal{C})} = \frac{W(\mathcal{D})}{W(\mathcal{C})}.$$

- Thus we have the ideal acceptance ratio $\alpha = 1$.
- Very efficient update: $\tau \sim L^{0.35}$.
- Only works with two-body interactions: $H = -\sum_{ij} J_{ij} s_i s_j$.
- U Wolff, PRL. **62**, 361 (1989).

Wolff algorithm: a cluster update

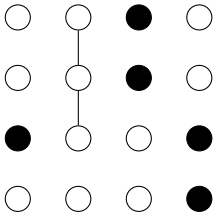


- A cluster is built one bond at a time.
- Probability of activating a bond is cleverly designed, such that

$$\frac{q(\mathcal{C} \to \mathcal{D})}{q(\mathcal{D} \to \mathcal{C})} = \frac{W(\mathcal{D})}{W(\mathcal{C})}.$$

- Thus we have the ideal acceptance ratio $\alpha = 1$.
- Very efficient update: $\tau \sim L^{0.35}$.
- Only works with two-body interactions: $H = -\sum_{ij} J_{ij} s_i s_j$.
- U Wolff, PRL. **62**, 361 (1989).

Wolff algorithm: a cluster update

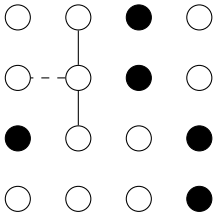


- A cluster is built one bond at a time.
- Probability of activating a bond is cleverly designed, such that

$$\frac{q(\mathcal{C} \to \mathcal{D})}{q(\mathcal{D} \to \mathcal{C})} = \frac{W(\mathcal{D})}{W(\mathcal{C})}.$$

- Thus we have the ideal acceptance ratio $\alpha = 1$.
- Very efficient update: $\tau \sim L^{0.35}$.
- Only works with two-body interactions: $H = -\sum_{ij} J_{ij} s_i s_j$.
- U Wolff, PRL. **62**, 361 (1989).

Wolff algorithm: a cluster update

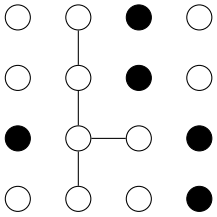


- A cluster is built one bond at a time.
- Probability of activating a bond is cleverly designed, such that

$$\frac{q(\mathcal{C} \to \mathcal{D})}{q(\mathcal{D} \to \mathcal{C})} = \frac{W(\mathcal{D})}{W(\mathcal{C})}.$$

- Thus we have the ideal acceptance ratio $\alpha = 1$.
- Very efficient update: $\tau \sim L^{0.35}$.
- Only works with two-body interactions: $H = -\sum_{ij} J_{ij} s_i s_j$.
- U Wolff, PRL. **62**, 361 (1989).

Wolff algorithm: a cluster update

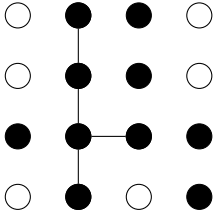


- A cluster is built one bond at a time.
- Probability of activating a bond is cleverly designed, such that

$$\frac{q(\mathcal{C} \to \mathcal{D})}{q(\mathcal{D} \to \mathcal{C})} = \frac{W(\mathcal{D})}{W(\mathcal{C})}.$$

- Thus we have the ideal acceptance ratio $\alpha = 1$.
- Very efficient update: $\tau \sim L^{0.35}$.
- Only works with two-body interactions: $H = -\sum_{ij} J_{ij} s_i s_j$.
- U Wolff, PRL. **62**, 361 (1989).

Wolff algorithm: a cluster update



- A cluster is built one bond at a time.
- Probability of activating a bond is cleverly designed, such that

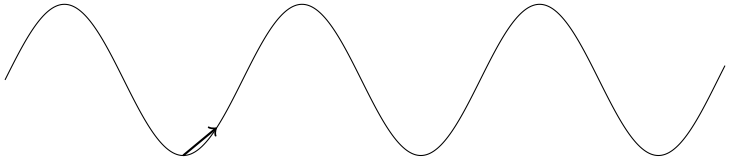
$$\frac{q(\mathcal{C} \to \mathcal{D})}{q(\mathcal{D} \to \mathcal{C})} = \frac{W(\mathcal{D})}{W(\mathcal{C})}.$$

- Thus we have the ideal acceptance ratio $\alpha = 1$.
- Very efficient update: $\tau \sim L^{0.35}$.
- Only works with two-body interactions: $H = -\sum_{ij} J_{ij} s_i s_j$.
- U Wolff, PRL. **62**, 361 (1989).

Challenge

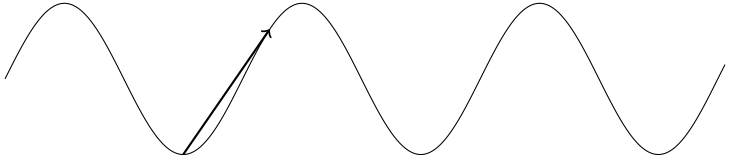
- Local update is too slow for many models: critical slowing down, glassy behavior, separation of energy scales, etc.
- Global update is only available for certain models. Like Wolff algorithm for two-body interactions.
- A good update algorithm for generic models?

A good update samples low-energy states



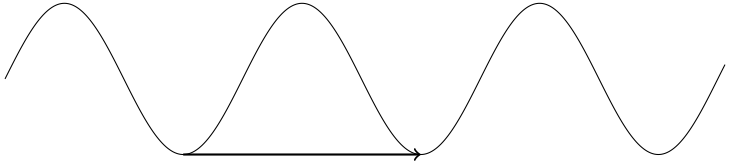
- A dilemma of local updates:
- Step is too small: high acceptance, small difference.
- Step is too big: low acceptance, big difference.
- Global updates: explore the low-energy configurations.

A good update samples low-energy states



- A dilemma of local updates:
- Step is too small: high acceptance, small difference.
- Step is too big: low acceptance, big difference.
- Global updates: explore the low-energy configurations.

A good update samples low-energy states



- A dilemma of local updates:
- Step is too small: high acceptance, small difference.
- Step is too big: low acceptance, big difference.
- Global updates: explore the low-energy configurations.

Modeling the update process

- A stochastic function $f : \mathcal{C} \mapsto \mathcal{D}$.
- Parameterized: $f = f(p_1, p_2, \ldots, p_n)$.
- Analytically known $q(\mathcal{C} \overset{f}{\mapsto} \mathcal{D})$.
- Optimize p_i, such that

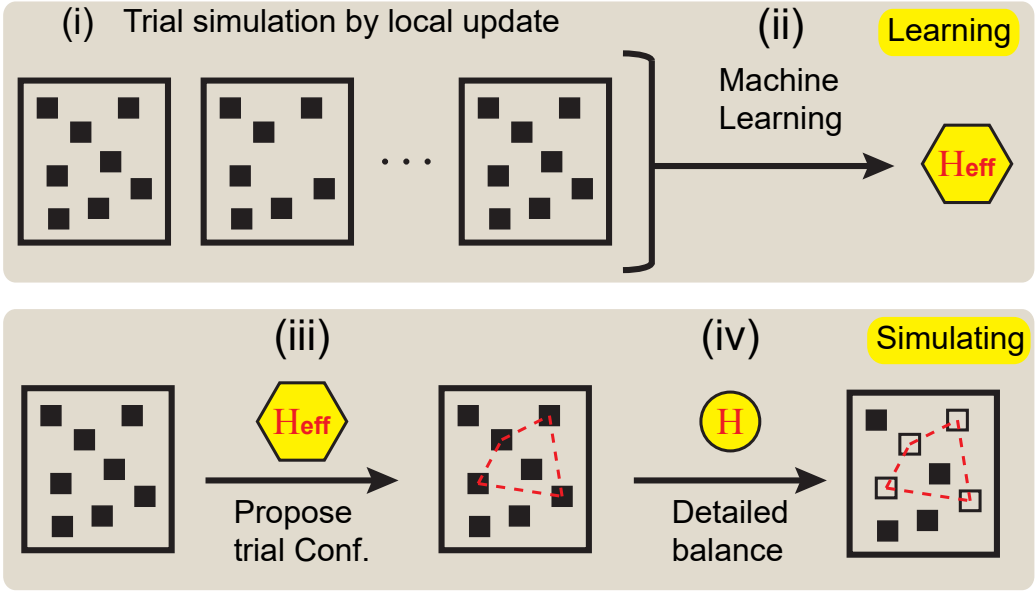
$$\frac{q(\mathcal{C} \overset{f}{\mapsto} \mathcal{D})}{q(\mathcal{D} \overset{f}{\mapsto} \mathcal{C})} \simeq \frac{W(\mathcal{D})}{W(\mathcal{C})}.$$

- Choose the acceptance ratio

$$\alpha(\mathcal{C} \to \mathcal{D}) = \min\left\{1, \frac{W(\mathcal{D})}{W(\mathcal{C})} \frac{q(\mathcal{D} \overset{f}{\mapsto} \mathcal{C})}{q(\mathcal{C} \overset{f}{\mapsto} \mathcal{D})}\right\} \simeq 1.$$

- The approximation $\frac{q(\mathcal{C} \overset{f}{\mapsto} \mathcal{D})}{q(\mathcal{D} \overset{f}{\mapsto} \mathcal{C})} \simeq \frac{W(\mathcal{D})}{W(\mathcal{C})}$ does not introduce error in the MC simulation, as long as we know $q(\mathcal{C} \overset{f}{\mapsto} \mathcal{D})$ accurately.

Design


(i) Trial simulation by local update
(ii)
Learning
Machine
Learning
H_{eff}
(iii)
H_{eff}
Propose
trial Conf.
(iv)
H
Detailed
balance
Simulating

Outline

Example:

- The model:
$$H = -J_1 \sum_{\langle ij \rangle} s_i s_j - J_2 \sum_{ijkl \in \square} s_i s_j s_k s_l.$$
- The effective model:
$$H_{\text{eff}} = -J_1^{\text{eff}} \sum_{\langle ij \rangle} s_i s_j.$$
- We consider $J_1 = 1$, $J_2 = 0.2$.
- An Ising transition at $T_c = 2.493$.
  Ising model without J_2 has $T_c = 2.269$.

The self-learning update

- The update: cluster is constructed using the effective model.

$$\frac{q(\mathcal{C} \to \mathcal{D})}{q(\mathcal{D} \to \mathcal{C})} = \frac{W_{\text{eff}}(\mathcal{D})}{W_{\text{eff}}(\mathcal{C})}.$$

- The acceptance ratio:

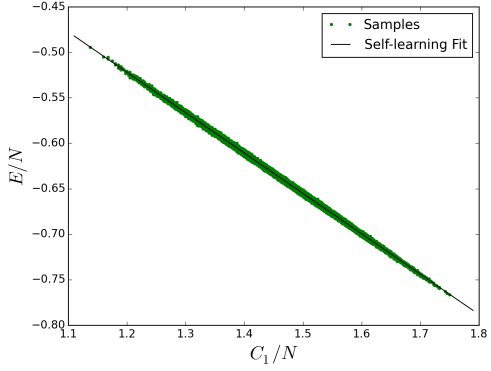
$$\alpha(\mathcal{C} \to \mathcal{D}) = \min\left\{1, \frac{W(\mathcal{D})}{W(\mathcal{C})} \frac{W_{\text{eff}}(\mathcal{C})}{W_{\text{eff}}(\mathcal{D})}\right\}.$$

- The algorithm still satisfies the detailed balance exactly, although the effective model is approximate.

Learning the effective model

1. Generate a sample using the local update.
2. Perform a linear regression,

$$H_{\text{eff}} = J_1^{\text{eff}} \sum_{\langle ij \rangle} S_i S_j + E_0.$$



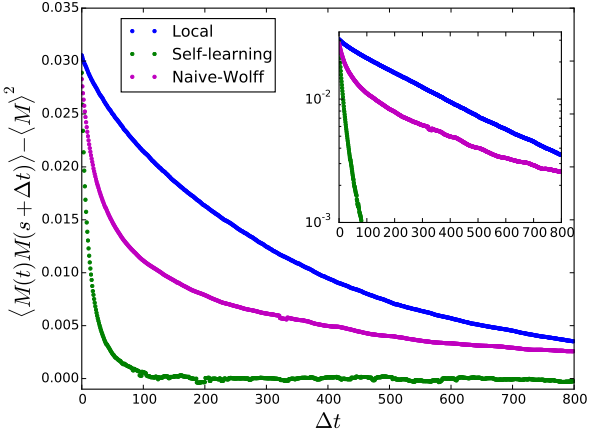
Unsupervised machine learning

- The linear regression we just did can be viewed as an unsupervised machine learning.
- Unsupervised ML: learning the underlying distribution from a sample of configurations.
- We pretend that we didn't know H, and learn a simpler model H_{eff}.
- The learned model describes the low-energy effective theory, where the high-energy fluctuations are treated as noise in the sample.
- Can incorporate more advanced ML models and algorithms.

Learning the effective model

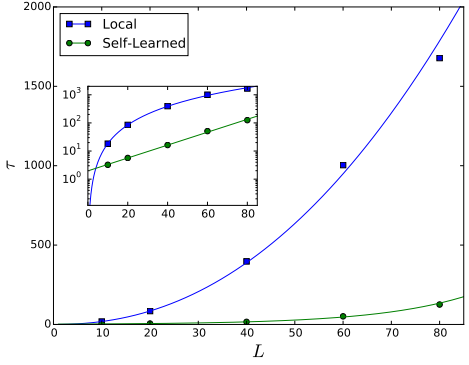
1. Generate a sample using the local update, at $T = 5 > T_c$.
2. Perform a linear regression,
$$H_{\text{eff}} = J_1^{\text{eff}} \sum_{\langle ij \rangle} S_i S_j + E_0.$$
3. $J_1^{\text{eff}} = 1.0726$.
4. Generate another sample using the self-learning update, at $T = T_c$.
5. $J_1^{\text{eff}} = 1.1064$.

Autocorrelation time



System size 40×40.

SLMC works well at moderate sizes

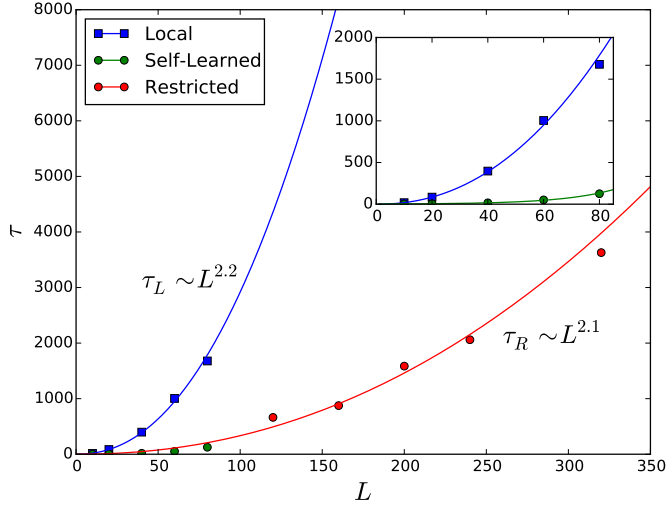


- The results look good at moderate sizes $L = 20 - 80$, but the scaling behavior is exponential...
- Eventually, the self-learning update becomes slower than the local update.

Limiting the cluster size

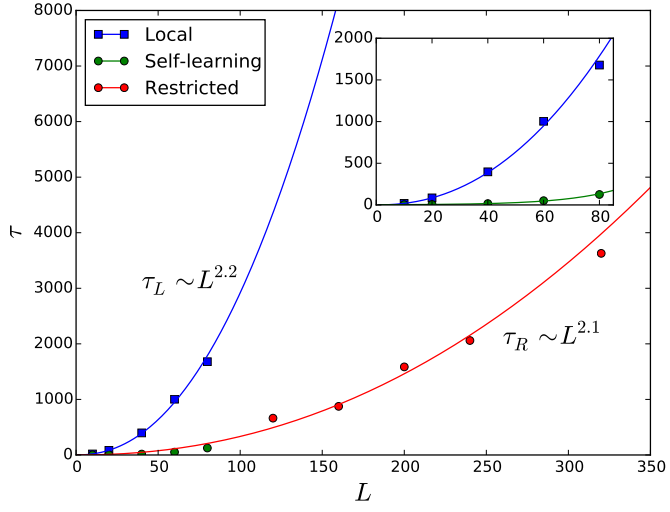
- $\Delta H = H - H_{\text{eff}}$ scales with the perimeter of the cluster $\sim L$.
- $\tau \sim e^{L/L_0}$ because $\alpha \sim e^{-\beta \Delta H} \sim e^{-L/L_0}$.
- A simple effective model is not accurate: need more accurate model, with a hierarchy of energy scales.
- Quick fix: limit the cluster size during cluster growing. Still satisfies the detailed balance, if α is corrected correspondingly.
- A block update: same scaling as the local update, but with a smaller prefactor.

Restricted-cluster-size update



We restrict the cluster size to $r = 40$.

Summary


Local
Self-learning
Restricted
$\tau_L \sim L^{2.2}$
$\tau_R \sim L^{2.1}$
τ
L
0
1000
2000
3000
4000
5000
6000
7000
8000
0
50
100
150
200
250
300
350
500
1500
20
40
60
80

Outline

Determinant Monte Carlo: computational cost

- Fermion coupling to (auxiliary) bosonic field.

$$H = -\sum_{ij} t_{ij} c_i^\dagger c_j + \sum_i \sigma_i c_i^\dagger c_i + H[\sigma].$$

- Simulation: use bosonic field as configurations.

$$\mathcal{C} = \{\sigma_i(t)\}, i = 1, \ldots N; 0 < t < \beta.$$

- Weight of each configuration $W(\mathcal{C})$: integrate out the fermions.
- Computational cost: βN^3.
- Fast update: βN^3 for βN steps of local updates.
- Generating two independent configurations: $\beta N^3 \tau_L$.

Cumulative update

- Idea: replace $W(\mathcal{C})$ by $W'(\mathcal{C})$, which is much faster to evaluate.
- Local updates guided by $W'(\mathcal{C})$:

$$\mathcal{C} \Rightarrow \mathcal{D} : \mathcal{C} \to \mathcal{C}_1 \cdots \to \mathcal{C}_{i-1} \to \mathcal{C}_i \to \mathcal{C}_{i+1} \to \cdots \to \mathcal{C}_M \to \mathcal{D}.$$

- Detailed balance:

$$\frac{p(\mathcal{C}_i \to \mathcal{C}_{i+1})}{p(\mathcal{C}_{i+1} \to \mathcal{C}_i)} = \frac{W(\mathcal{C}_{i+1})}{W(\mathcal{C}_i)}$$

- For each path:

$$\frac{p(\mathcal{C} \to \mathcal{C}_1) \cdots p(\mathcal{C}_i \to \mathcal{C}_{i+1}) p(\mathcal{C}_M \to \mathcal{D})}{p(\mathcal{C}_1 \to \mathcal{C}) \cdots p(\mathcal{C}_{i+1} \to \mathcal{C}_i) p(\mathcal{D} \to \mathcal{C}_M)} = \frac{W'(\mathcal{C}_1)}{W'(\mathcal{C})} \frac{W'(\mathcal{C}_2)}{W'(\mathcal{C}_1)} \cdots \frac{W'(\mathcal{C}_M)}{W'(\mathcal{C}_{M-1})} \frac{W'(\mathcal{D})}{W'(\mathcal{C}_M)} = \frac{W'(\mathcal{D})}{W'(\mathcal{C})}.$$

- Summing over all paths:

$$\frac{q(\mathcal{C} \to D)}{q(\mathcal{D} \to \mathcal{C})} = \frac{p(\mathcal{C} \Rightarrow D)}{p(\mathcal{D} \Rightarrow \mathcal{C})} = \frac{\sum_{\mathcal{C}_i} p(\mathcal{C} \to \mathcal{C}_1) \cdots p(\mathcal{C}_i \to \mathcal{C}_{i+1}) p(\mathcal{C}_M \to \mathcal{D})}{\sum_{\mathcal{C}_i} p(\mathcal{C}_1 \to \mathcal{C}) \cdots p(\mathcal{C}_{i+1} \to \mathcal{C}_i) p(\mathcal{D} \to \mathcal{C}_M)} = \frac{W'(\mathcal{D})}{W'(\mathcal{C})}.$$

Cumulative update

- Approximated weight:
$$W'(\mathcal{C}) \simeq W(\mathcal{C}).$$
- Update scheme:
$$\frac{q(\mathcal{C} \to D)}{q(\mathcal{D} \to \mathcal{C})} = \frac{W'(\mathcal{D})}{W'(\mathcal{C})} \simeq \frac{W(\mathcal{D})}{W(\mathcal{C})}.$$
- Acceptance ratio:
$$\alpha(\mathcal{C} \to \mathcal{D}) = \min\left\{1, \frac{W(\mathcal{D})}{W(\mathcal{C})}\frac{W'(\mathcal{C})}{W'(\mathcal{D})}\right\} \simeq 1.$$
- Choose good W' and $M \geq \tau_L$: $\tau_{CU} \sim 1$.
- Computational cost: $\beta N^3 \tau \to \beta N^3$.
- Again, $W'(\mathcal{C}) \neq W(\mathcal{C})$ does not introduce approximation.

Constructing effective weights

1. Physical intuition: bosonic effective model of $W_{\text{eff}}[\sigma]$ + Linear Regression.
   Works best for gapped fermions.

$$H = -\sum_{ij} t_{ij} c^{\dagger}_{i\alpha} c_{j\alpha} + \sum_{i} \sigma_i c^{\dagger}_{i\alpha} \sigma^{z}_{\alpha\beta} c_{i\beta} + H[\sigma]$$

$$W_{\text{eff}}[\sigma] = \sum_{ij} J_{ij} \sigma_i \sigma_j .$$

2. Machine Learning models: Restricted Boltzmann Machine, Neural Networks, etc.

Self-Learning MC in action

- Zi-Hong Liu, Xiao-Yan Xu, Yang Qi, Kai Sun and Zi-Yang Meng, to appear.
- Fermion + boson model, new critical universality. Simulated at the critical point.
- SLMC: $L = 20 \rightarrow 30$.