Statistics and Computing (2019) 29:1035–1053
https://doi.org/10.1007/s11222-019-09852-6

Convergence analysis of herded-Gibbs-type sampling algorithms: effects of weight sharing

Hiroshi Yamashita[1] · **Hideyuki Suzuki**[2]

Received: 18 November 2017 / Accepted: 7 January 2019 / Published online: 22 January 2019

Abstract
Herded Gibbs (HG) and discretized herded Gibbs (DHG), which are Gibbs samplings combined with herding, are deterministic sampling algorithms for Markov random fields with discrete random variables. In this paper, we introduce the notion of "weight sharing" to systematically view these HG-type algorithms, and also investigate their convergence theoretically and numerically. We show that, by sharing and reducing the number of weight variables, the HG-type algorithm achieves fast initial convergence at the expense of asymptotic convergence. This means that the HG-type algorithm can be practically more efficient than conventional Markov chain Monte Carlo algorithms, although its estimate does not necessarily converge to the target asymptotically. Moreover, we decompose the numerical integration error of HG-type algorithms into several components and evaluate each of them in relation to herding and weight sharing. By using this formulation, we also propose novel variants of the HG-type algorithm that reduce the asymptotic bias.

Keywords Herded Gibbs · Herding · Markov chain Monte Carlo · Deterministic sampling · Boltzmann machine

1 Introduction

The Markov chain Monte Carlo (MCMC) algorithm is widely used in the fields of statistics and machine learning to estimate an expectation efficiently in a high-dimensional probability space. The estimate obtained by MCMC is theoretically guaranteed to converge to the target asymptotically—in the limit where the number of samples T goes to infinity.

In practice, however, we need an estimate within a limited computation time for solving real-world problems. In other words, we have to stop generating samples at finite T and accept a certain amount of estimation error. In particular, when the probability space is high dimensional, the sample sequence that we can utilize must be very short in comparison with its large sample space. What we really need the most in practice is an MCMC algorithm that can reduce the estimation error to an acceptable level at the earliest possible time. In this sense, it is not only asymptotic convergence that is important.

Herding (Welling 2009a, b; Chen et al. 2010) is a deterministic sampling algorithm implemented as a weakly chaotic dynamical system. It produces a sample sequence such that the moments of features converge to the target values with the prominent convergence rate $O(1/T)$. That is considerably faster than the rate $O(1/\sqrt{T})$ of random sampling algorithms. However, we can only specify the moments in herding; in other words, herding is not suitable for a case that we need a sample sequence from a predefined target distribution.

Herded Gibbs (HG) (Chen et al. 2016) is a deterministic sampling algorithm for Markov random fields (MRFs) that is based on herding. It is designed to be used instead of random Gibbs sampling and seems to inherit the fast convergence of the herding. Discretized herded Gibbs (DHG) (Eskelinen 2013) is a variant of HG that reduces the large spatial complexity of HG.

The superiority of these HG-type algorithms to Gibbs sampling is affirmed empirically. It is also theoretically shown that the convergence rate is $O(1/T)$ for the MRF on a fully connected graph. However, there is no guarantee for general cases. It is also reported that HG has an estima-

The codes used for the experiments in this paper are available in https://github.com/utatakiyoshi/Herded_Gibbs_STCO.

✉ Hiroshi Yamashita
hiroshi_yamashita@mist.i.u-tokyo.ac.jp

[1] Graduate School of Information Science and Technology, The University of Tokyo, Tokyo, Japan

[2] Graduate School of Information Science and Technology, Osaka University, Osaka, Japan

tion bias in some cases, but this has not yet been investigated theoretically. DHG also has a greater estimation bias than HG.

Because we can only utilize a limited number of samples in practice, the bias is not a problem as far as it is less significant than the unavoidable variance caused by the limitation of sample length. To pursue the capabilities and mitigate the drawbacks of HG, we need an analysis of the convergence behavior in general cases that do not have a mathematical guarantee. Moreover, if we understand HG and DHG in a unified manner, it will help in developing new variants of HG.

In this paper, we investigate the convergence behavior of HG by numerical observation and mathematical formulation. Rather than investigating only HG, we combine it with DHG as HG-type algorithms. Based on the study, we also discuss when and how to use HG-type algorithms.

The remainder of this paper is organized as follows. After an introduction of existing HG-type algorithms in Sect. 2, we introduce the notion of "weight sharing" in Sect. 3. We discuss the effects of "weight sharing" and present a schematic of the convergence behavior with a numerical example. In Sect. 4, we give a mathematical formulation of the above discussion and make an upper bound of estimation error. In Sects. 5 and 6, we apply the proposed formulation to a case of MRFs with binary variables. Proposals of new HG-type algorithms and comparisons between proposed variants and existing methods are also made in both sections. In Sect. 7, we use image reconstruction as an example of the task where the estimation error with small sample size is more important than the asymptotic convergence, and show HG-type algorithms perform well for this task by a numerical experiment. In Sect. 8, finally, we give some concluding remarks.

Deterministic Markov Chains, which include HG-type algorithms, have a long history of research. Most importantly, QMCMC (Chentsov 1967; Tribble 2007) is a combination of quasi Monte Carlo and MCMC, which uses a deterministic driving sequence for MCMC. The driving sequence has a "balance in column" (Tribble 2007), which means that the column of the matrix consisting of row vectors of driving numbers is used for updates of MC steps. For Gibbs sampling, each column corresponds to a variable and the numbers for updating each variable distribute equally. Thus, the estimation variances for variables are reduced. It also preserves the asymptotic consistency by making consecutive numbers nearly independent for each variable, and the consistency is guaranteed by the characteristics of WCUD (weakly completely uniformly distributed). QMCMC also has been studied by theoretical analysis including the asymptotic consistency (Chen et al. 2011) and error bounds (Dick et al. 2016). In Sect. 7, we compared the performance of HG-type algorithms with QMCMC, considering its importance described here.

2 Herding and Herded-Gibbs-type algorithms

In this section, we introduce existing HG-type algorithms and some preliminary knowledge.

2.1 Gibbs sampling and Boltzmann machine

Gibbs sampling is one sort of MCMC method. Let $\pi(x_1, \ldots, x_N)$ be a MRF of N random variables. The conditional distribution of x_i given the other variables is known for all x_i. Gibbs sampling generates samples from π by updating variables one by one using the conditional distributions.

A Boltzmann machine is one of the MRFs to which Gibbs sampling is often applied. Let $x_1, \ldots, x_N$ be random variables taking values in $\{0, 1\}$, and let (V, E) be the undirected graph that represents the dependency of the variables, where $V = \{1, \ldots, N\}$. In a Boltzmann machine, the joint distribution is expressed by

$$\pi(x_1, \ldots, x_N) = \frac{1}{Z} \exp\left(-H(x_1, \ldots, x_N)\right),$$
$$H(x_1, \ldots, x_N) = -\sum\nolimits_{i\in V} B_i x_i - \sum\nolimits_{(i,j)\in E} W_{ij} x_i x_j,$$

where B_i and W_{ij} are called the bias parameter and coupling parameter, respectively, and

$$Z = \sum\nolimits_{x_1,\ldots,x_N\in\{0,1\}} \exp(-H(x_1, \ldots, x_N))$$

is a normalizing constant to make the sum of probability one, which is called the partition function.

For each ith variable x_i, let $\mathcal{N}(i) = \{j | (i, j) \in E\}$ be the set of variables that are neighboring to x_i, and let $V_{-i} = V \backslash \{i\}$ be the indices of all variables except x_i. Let us denote the state of $\mathcal{N}(i)$ and V_{-i} by $\boldsymbol{x}_{\mathcal{N}(i)}$ and $\boldsymbol{x}_{-i}$, respectively. In a Boltzmann machine, the conditional distribution is expressed by a simple form as

$$\begin{aligned}\pi(x_i|\boldsymbol{x}_{-i}) &= \pi(x_i|\boldsymbol{x}_{\mathcal{N}(i)}) \\ &= \frac{1}{Z_i(\boldsymbol{x}_{\mathcal{N}(i)})} \exp\left(x_i\left(B_i + \sum\nolimits_{j\in\mathcal{N}(i)} W_{ij} x_j\right)\right),\end{aligned}$$

where $Z_i(\boldsymbol{x}_{\mathcal{N}(i)})$ is a normalizing factor. Thus, the sampling from a Boltzmann machine can be done by Gibbs sampling as shown in Algorithm 1.

Please note that, in the following part, we sometimes use Boltzmann machines with binary variables taking values in $\{-1, +1\}$ instead of $\{0, 1\}$, which will be stated explicitly.

2.2 Herding

Herding (Welling 2009a, b; Chen et al. 2010) is a nonlinear dynamical system for generating a sequence of pseudo-

Algorithm 1 Gibbs sampling for a Boltzmann Machine

```
Initialize $\{x_i\}$.
for $t = 1$ to $T$ do
    for $i = 1$ to $N$ do
        Sample $x_i$ from the distribution $\pi(x_i | \boldsymbol{x}_{\mathcal{N}(i)})$.
    end for
    $\boldsymbol{x}^{(t)} \leftarrow (x_1, \ldots, x_N)$.
end for
```

samples from a finite sample space X. Let $\boldsymbol{\phi} : X \to \mathbb{R}^n$ be a deterministic function (feature map) and $\boldsymbol{\mu} \in \mathbb{R}^n$ be a constant vector (target moment vector of the features). In herding, the sample point $x^{(t)}$ is updated alternately with the weight vector $\boldsymbol{w}^{(t)} \in \mathbb{R}^n$ as follows:

$$x^{(t)} = \mathrm{argmax}_{x \in X} \langle \boldsymbol{w}^{(t-1)}, \boldsymbol{\phi}(x) \rangle,$$
$$\boldsymbol{w}^{(t)} = \boldsymbol{w}^{(t-1)} + \boldsymbol{\mu} - \boldsymbol{\phi}(x^{(t)}).$$

To visit all the points in X, $\boldsymbol{\phi}$ should map X into a set of points that are extreme points of a convex set in $\mathbb{R}^n$.

Owing to the weakly chaotic behavior of $\boldsymbol{w}^{(t)}$, the generated sequence $x^{(t)}$ appears random, while the procedure is deterministic. Moreover, the empirical moment $\hat{\boldsymbol{\mu}} = (1/T) \sum_t \boldsymbol{\phi}(x^{(t)})$ matches $\boldsymbol{\mu}$ with an error of $O(1/T)$, which is considerably smaller than $O(1/\sqrt{T})$ of random sampling. This procedure can be interpreted as greedily minimizing the error between $\hat{\boldsymbol{\mu}}$ and $\boldsymbol{\mu}$ (Welling and Chen 2010).

Unlike i.i.d. random sampling, the samples generated by herding are not independent of each other. For example, consider a case in which a feature of a drawn sample $\phi_i(x)$ is greater than the target moment μ_i. Then, the corresponding weight decreases in the update; hence, samples with large $\phi_i(x)$ are less likely to be drawn in the following steps. Therefore, the sample sequence has a negative auto-correlation, which helps the quick moment-matching property of herding (Chen et al. 2010).

It is only the moments where the sequence is generated by herding matches, and herding is not a sampling algorithm from a given distribution. However, it can be used for sampling from Bernoulli distributions and multinomial distributions as follows.

Let us consider a Bernoulli case where the sample space is $X = \{0, 1\}$, the feature map is $\phi(x) = x$, and thus the target moment and weight are scalar. In this case, the distribution is completely determined by the mean of the feature $p \equiv E[x]$. The herding procedure for this case is given in Algorithm 2. The sample $x^{(t)}$ is determined only by the sign of the weight $w^{(t-1)}$ because it holds that $\mathrm{argmax}_{x \in \{0,1\}} wx = \mathbb{I}(w > 0)$. It also holds that $w^{(T)} - w^{(0)} = Tp - \sum_t x^{(t)}$. Because it can be shown that $w^{(t)}$ is bounded, the empirical moment of the samples is generated by herding converges at the rate of $O(1/T)$ as $\left|p - \sum_t x^{(t)}/T\right| = \left|w^{(T)} - w^{(0)}\right|/T$. This procedure can be used as a deterministic alternative to random sampling from the Bernoulli distribution $Ber(p)$.

In the case of a multinomial distribution $\pi(X = i) = p_i$, herding can be derived by setting the feature as $\phi_i(x) = \mathbb{I}(x = i)$ and the target moment as $\mu_i = p_i$ (Algorithm 3).

Algorithm 2 Herding for a Bernoulli distribution

```
$x^{(t)} \leftarrow \mathbb{I}(w^{(t-1)} > 0)$.
$w^{(t)} \leftarrow w^{(t-1)} + p - x^{(t)}$.
```

Algorithm 3 Herding for a multinomial distribution

```
$x^{(t)} \leftarrow \mathrm{argmax}_x w_x^{(t-1)}$.
$w_x^{(t)} \leftarrow w_x^{(t-1)} + p_x - \mathbb{I}(x^{(t)} = x)$, for all $x$.
```

2.3 Herded Gibbs

Herded Gibbs (HG) (Chen et al. 2016) is a deterministic sampling algorithm for a Boltzmann machine that is designed to be used instead of Gibbs sampling. In HG, random sampling from a conditional distribution is replaced by the sampling using herding.

Let us consider a Boltzmann machine of N variables which take values in $\{0, 1\}$. The conditional probability distribution of x_i given other variables is determined only by the neighboring variables, i.e., $\pi(x_i | \boldsymbol{x}_{-i}) = \pi(x_i | \boldsymbol{x}_{\mathcal{N}(i)})$. This is a Bernoulli distribution $Ber(p_{i,y})$ where $p_{i,y} = \pi(x_i = 1 | \boldsymbol{x}_{\mathcal{N}(i)} = y)$. Thus, x_i is updated by a sample from $Ber(p_{i,y})$ in Gibbs sampling.

By replacing the random sampling with the herding, which is shown in Algorithm 2, we obtain HG. The overall procedure of HG is shown in Algorithm 4 (cf. Algorithm 1). It should be noted that we use one weight variable $w_{i,y}$ for each conditioning state y. Therefore, each variable x_i has $2^{|\mathcal{N}(i)|}$ weight variables.

The initial values for the weight variables $w_{i,y}$ can be chosen arbitrarily. However, in practice, a systematic initialization, such as $w_{i,y} = 0$ for all i and y, may cause a severe bias at the early stage of the sampling. Therefore, it is better to draw $w_{i,y} \in (p_{i,y} - 1, p_{i,y}]$ uniformly at random. With this

Algorithm 4 Herded Gibbs for a Boltzmann machine

```
Initialize $\{x_i\}$ and $\{w_{i,y}\}$.
for $t = 1$ to $T$ do
    for $i = 1$ to $N$ do
        Let $y = \boldsymbol{x}_{\mathcal{N}(i)}$, which is the state of the neighbors of $x_i$.
        $x_i \leftarrow \mathbb{I}(w_{i,y} > 0)$.
        $w_{i,y} \leftarrow w_{i,y} + \pi(x_i = 1 | \boldsymbol{x}_{\mathcal{N}(i)}) - x_i$.
    end for
    $\boldsymbol{x}^{(t)} \leftarrow (x_1, \ldots, x_N)$.
end for
```

Algorithm 5 Updating x_i in discretized herded Gibbs

```
Calculate the conditional probability and let $p = \pi(x_i = 1|y)$,
Find an integer $b$ such that $\theta_b < p \le \theta_{b+1}$.
$x_i \leftarrow \mathbb{I}(w_{i,b} > 0)$.
$w_{i,b} \leftarrow w_{i,b} + p - x_i$.
```

initialization, $w_{i,y}$ is kept in the same range $(p_{i,y} - 1, p_{i,y}]$ during the algorithm.

Theoretically, the convergence of the estimation of HG is shown for only two special types of graphs: empty graphs and fully connected graphs (Chen et al. 2016). For fully connected graphs, HG is guaranteed to achieve an $O(1/T)$ convergence rate. However, in general cases not covered by the theory, a simple example of an incomplete graph for which HG does not converge is reported in their paper. Nevertheless, the performance of HG when applied to some machine learning tasks is better than that of conventional Gibbs sampling and of several mean field methods (Chen et al. 2016). Note that the running time of HG per iteration is almost the same as that of Gibbs sampling.

HG can deal with MRFs with not just binary variables but also discrete variables by replacing the herding step for a Bernoulli distribution (Algorithm 2) with that for a multinomial distribution (Algorithm 3).

2.4 Discretized herded Gibbs

Discretized herded Gibbs (DHG) (Eskelinen 2013) is a variant of HG, in which the number of weight variables is reduced. In DHG, the range of probabilities [0, 1] is divided into B segments with $0 = \theta_0 < \theta_1 < \cdots < \theta_{B-1} < \theta_B = 1$. Each bin $(\theta_b, \theta_{b+1}]$ has a single weight variable $w_{i,b}$ for each ith variable. The partition can be set arbitrarily. However, for the sake of simplicity, we choose a uniform division with the width of $1/B$ for all variables in the following. Algorithm 5 shows how DHG draws a sample from a conditional distribution. The dynamics of a weight variable is not exactly equivalent to herding dynamics because the moment parameter p differs in every update. However, if these parameters p are in a sufficiently close interval, the dynamics is close to herding dynamics up to the limited accuracy. The accuracy is governed by the number of bins B. It has been empirically shown that at the initial stage of sampling, the distribution of the generated samples exhibits fast convergence to the target distribution, but the convergence stops eventually (Eskelinen 2013).

3 Weight sharing and convergence behavior

In this section, we present a view on the convergence behavior of HG-type algorithms with the notion of "weight sharing." We also investigate the difference of the behaviors between these algorithms and conventional Gibbs sampling in relation to the proposed view.

3.1 Weight sharing

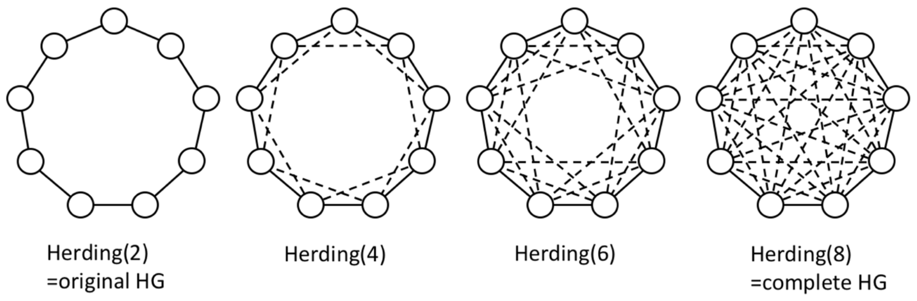
As discussed in the previous section, HG is not generally assured to yield samples that converge to the target distribution; the asymptotic convergence is only guaranteed for MRFs on fully connected graphs. However, it is true that any MRF, which is usually defined on an incomplete graph, can be virtually considered to be defined on a fully connected graph by adding extra edges that have no effect on the distribution. In the Boltzmann machine case, the extra edges have coupling parameter $W_{ij} = 0$. HG on this graph is guaranteed to converge asymptotically as $O(1/T)$, while it loses the practical utility because of the significant increase in spatial complexity.

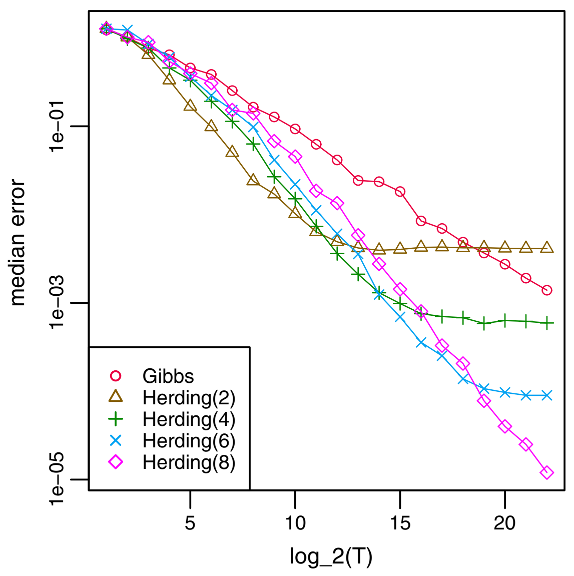
In complete HG, the index of weight variable to be used is determined by a full conditioning state $\boldsymbol{x}_{-i}$ that includes all the ineffective variables in addition to the original neighbors. Let us denote this index by z.

A set of full conditioning states $\{z\}$ that have the same state of $\boldsymbol{x}_{\mathcal{N}(i)}$ corresponds to one partial conditioning state y. Therefore, the set of weight variables $\{w_{i,z}\}$ in the complete HG is aggregated into one weight variable $w_{i,y}$ in the original HG. In other words, a weight variable in HG is shared by multiple full conditioning states. DHG can also be viewed similarly; in DHG, more full conditioning states, whose conditional probability values are in the same bin $[\theta_b, \theta_{b+1}]$, share a single weight variable $w_{i,b}$. Therefore, exploring the effect of weight sharing allows us to understand these HG-type algorithms in a consistent manner.

Let $\hat{p}_{i,z}$ be the ratio of the number of times that a sample $x_i = 1$ is generated when $\boldsymbol{x}_{-i} = z$ is satisfied on the update of x_i. It is the ratio of 1's in the sample sequence generated by $w_{i,z}$, and converges to $\pi(x_i = 1 \mid \boldsymbol{x}_{-i} = z)$ at the rate of $O(1/T)$ as explained previously. This allows the empirical transition matrix to converge to the transition matrix of Gibbs sampling and therefore guarantees the asymptotic convergence to the target MRF with $O(1/T)$. It is the core of the proof for the asymptotic convergence of the complete HG in Theorem 3 of Chen et al. (2016).

By the theorem, the convergence is guaranteed only for complete HG. In a general case of HG where a weight variable $w_{i,y}$ is shared by multiple full conditioning states z, the sample sequence convergence of $w_{i,y}$ does not necessarily mean the convergence of $\hat{p}_{i,z}$ for each z. Therefore, the convergence of the transition matrix and the asymptotic convergence of HG does not necessarily hold in this case, as it is also shown by the counterexample in Chen et al. (2016).





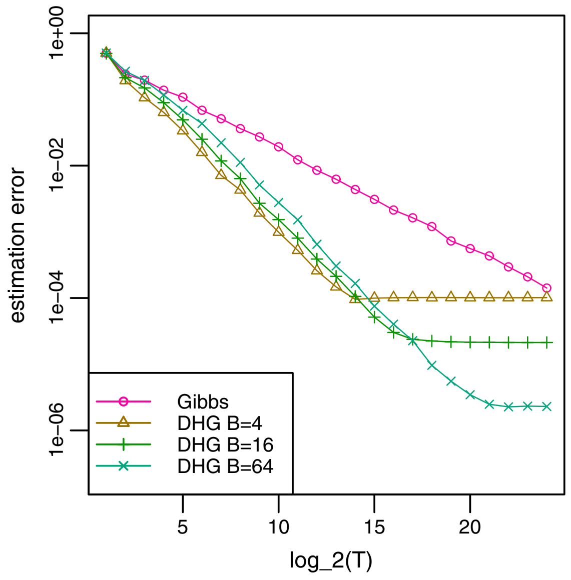


Fig. 1 Top: Boltzmann machines that are used in the experiment in Sect. 3.2. The dashed lines denote the dummy edges that have a coupling parameter $W_{ij} = 0$. The leftmost one is according to HG, and the rightmost one is according to complete HG. The other two interpolate between the two algorithms by sharing weight variables partially from the complete HG. Bottom left: median error of 100 estimates of $E[\sum_i x_i]$ for HGs above and Gibbs sampling. Bottom right: estimation error of $E[x_8]$ in an $N = 8$ fully connected Boltzmann machine. The absolute errors of estimation for 100 times are averaged. The parameters are drawn from $[0, 0.1]$

3.2 Effects of weight sharing

How does "weight sharing" work in HG? By carefully observing the behavior of HG-type algorithms, we found two effects of weight sharing. We describe them in relation to the following example.

Consider a Boltzmann machine with $N = 9$ variables which take values in $\{-1, +1\}$. The variables are lined up in a ring and are connected by the nine edges represented by $(x_i, x_{i+1 \bmod 9})$. The coupling parameters W_{ij} are taken independently from the normal distribution $\mathcal{N}(0.5, 0.05)$, and the biases B_i are taken independently from the normal distribution $\mathcal{N}(0.2, 0.05)$.

We apply Gibbs sampling and the following four types of HG as shown in the top panel of Fig. 1. The first is the original HG, and each variable has 2^2 weight variables. The second is HG with nine dummy edges represented by $(x_i, x_{i+2 \bmod 9})$. For one variable, there are four adjacent variables, so the number of weight variables per variable is 2^4. We added dummy edges represented by $(x_i, x_{i+3 \bmod 9})$ to obtain the third algorithm, and further added those represented by $(x_i, x_{i+4 \bmod 9})$ to obtain the fourth algorithm. The fourth algorithm is a HG for the complete graph, and consistency is guaranteed. These four types of HG are expressed as Herding(2), …, Herding(8) using the number of adjacent variables.

Using the five algorithms, we calculated an expected value $E[\sum x_i]$, 100 times each. In each execution, we set randomly the initial values of the weight variable and spin configuration. The bottom left panel of Fig. 1 shows the median, for each algorithm, of the absolute errors of the estimates. The true expectation is calculated directly according to the definition of the Boltzmann machine.

The first effect of weight sharing is the asymptotic bias of the estimation. In HG, the weight variable to be used is determined by the state of neighbors $\mathcal{N}(i)$. Therefore, x_i is drawn according to the state of only $\mathcal{N}(i)$. Thus, x_i and the variables of $V_{-i} \backslash \mathcal{N}(i)$ are expected to be independent given

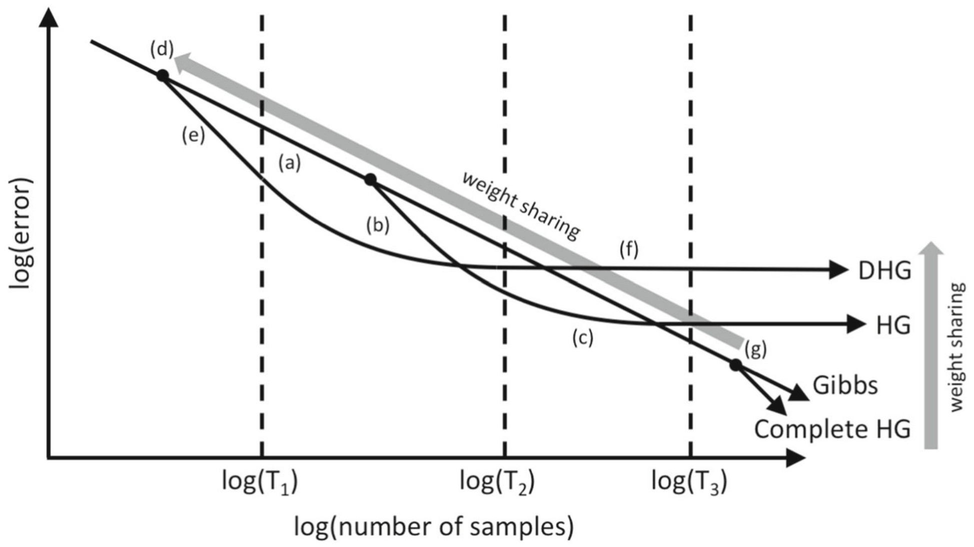


Fig. 2 Schematic view of the convergence behaviors of Gibbs sampling, complete HG, HG, and DHG

$\boldsymbol{x}_{\mathcal{N}(i)}$ in the sample distribution. For the target MRF, they are exactly independent. However, this does not hold for the sample distribution. Samples of x_i have negative auto-correlation because of herding. The states of $V_{-i} \backslash \mathcal{N}(i)$ are also temporally correlated. These temporal correlations cause a spatial correlation between x_i and the variables of $V_{-i} \backslash \mathcal{N}(i)$ in samples.

In the bottom left panel of Fig. 1, the error of Gibbs and Herding(8), which are guaranteed to be consistent, monotonically decreases. However, the error reduction of Herding(2), Herding(4), and Herding(6), where the weights were shared, stopped halfway, and converged to the value including the bias. In the case of Herding(2), where more weights were shared, the final error is greater.

Second, weight sharing affects the behavior in the initial period of iterations. When the number of weight variables is large, there is a long interval between updates of the same weight variable. Then, in the initial period that is relatively short to this interval, almost every weight variable is used at most only once. Then, there is no difference between HG and Gibbs sampling in this period, because we initialize the weights randomly to avoid the initial bias. After this Gibbs-like period, HG outperforms Gibbs sampling because the weights are updated more than once and the effect of herding begins to work. The length of this Gibbs-like period depends on the number of the weight variables. Weight sharing reduces the number of weight variables and shortens the Gibbs-like period, because the weights are consequently updated more often. Thus, weight sharing not only reduces the spatial complexity, but also boosts the initial performance of HG-type algorithms.

In the bottom left panel of Fig. 1, since Herding(8) is a HG on the complete graph, it converges as $O(1/T)$, but the difference from Gibbs becomes apparent only after $\log_2(T) = 8$. In the initial $\log_2(T) < 10$ region, Herding(2) gives more accurate estimation.

From these observations, it is suggested that "weight sharing" has a trade-off between the speed of convergence at the early iteration and the bias of the final convergence. Note that here we do not approximate the herding dynamics as in DHG, so that these effects are purely due to "weight sharing."

The bottom right panel of Fig. 1 shows the result of an experiment for DHG. If we increase B, less weights are shared and the degree of weight sharing decreases. The change in the curve by decreasing the degree of weight sharing behaves similarly to that of the bottom left panel.

3.3 Schematic interpretation of convergence behavior

Figure 2 illustrates our view on the convergence behavior of HG-type algorithms.

First, Gibbs sampling converges as $O(1/\sqrt{T})$. HG converges in the same manner as Gibbs sampling initially. The length of this Gibbs-like period (a) corresponds to the number of weights, which increases exponentially with the maximum degree of the graph and depends on how the variables depend on each other. Then, HG converges faster than Gibbs sampling (b). Finally, HG ends up with an asymptotic bias (c). Similarly, DHG behaves in accordance with Gibbs sampling initially (d), converges faster (e), and eventually reaches the asymptotic bias (f). However, compared with HG, in this case, the length of (d) is shorter and the asymptotic bias is greater. The complete HG also has "Gibbs-like" period, but its length (g) is longer than HG and in the exponential order of the size of the graph.

The effects of weight sharing are denoted by gray arrows. When the degree of weight sharing is increased from complete HG through HG to DHG, the length of initial Gibbs-like period decreases and the asymptotic bias increases.

Here, we find the possible superiorities of HG-type algorithms in practice. Recall that we have to stop sampling at a finite number of samples T. If we stop at T_1, the initial fast convergence of DHG works, whereas HG exhibits no difference from Gibbs sampling. As for T_2, DHG ceases to converge and HG achieves the most accurate estimation. If we continue sampling until T_3, even HG reaches the asymptotic bias and only Gibbs sampling and complete HG continues to converge thereafter. Thus, with an appropriate selection of the algorithm and its parameter values, we can expect HG or DHG to perform better than Gibbs sampling for realistic computation time $T \ll T_3$.

4 Decomposition of numerical integration error of HG-type algorithms

So far, we have observed the qualitative effects of weight sharing on the convergence behavior of the HG-type algorithms. In this section, we propose a mathematical formulation for the errors of HG-type algorithms for more concrete and detailed analysis.

4.1 Unified formulation of HG-type algorithms

First, we introduce some variables and notation that consistently represent HG-type algorithms including HG, complete HG, and DHG.

It is not straightforward to analyze the sampling dynamics of HG as a whole. However, we can say that the sample sequence generated by a single weight variable $w_{i,y}$ approximates $\pi(x_i \mid \boldsymbol{x}_{\mathcal{N}(i)} = y)$. Therefore, we fix one variable x_i to be focused and handle the conditional distributions $P(x_i \mid \boldsymbol{x}_{\mathcal{N}(i)} = y)$ for every y separately from the distribution of condition occurrence $P(\boldsymbol{x}_{\mathcal{N}(i)})$.

Let us denote the focused variable x_i by X. Then, denote the state of the remaining variables by Z. In other words, $Z = \boldsymbol{x}_{-i}$, which denotes the full conditioning state. Then, we can take out the conditional distribution of the focused X from the joint probability as

$$\pi(X, Z) = \pi(X|Z)\pi(Z).$$

All the HG-type algorithms have almost the same structure. However, the way of sharing weight variables differs depending on the algorithm. We describe the selection of the herding weight variable as $\pi(Y|Z)$, where Y is a random variable that indicates the weight variable to be used. Algorithm 6 shows the sampling procedure of HG-type algorithms in this formulation. In complete HG, because the weight variable is not shared and selected according to a full conditioning state,

$$\pi(Y|Z) = \mathbb{I}(Y = \boldsymbol{x}_{-i}) = \mathbb{I}(Y = Z).$$

In original HG, the weight variable is shared and selected according to the state of variables in $\mathcal{N}(i)$. Then,

$$\pi(Y|Z) = \mathbb{I}(Y = \boldsymbol{x}_{\mathcal{N}(i)}).$$

In DHG, the variable Y represents the index b of the bin satisfying the condition $\pi(X = 1|Z) \in (\theta_b, \theta_{b+1}]$, and

$$\pi(Y = b|Z) = \mathbb{I}[\pi(X = 1|Z) \in (\theta_b, \theta_{b+1})].$$

In these examples, the conditional distribution functions $\pi(Y|Z)$ take a value of 0 or 1, which means that Y is chosen deterministically according to Z. However, in this formulation Y can be chosen randomly as we consider later (Sect. 6.5).

In the following, we denote the value of X, Y, and Z by x, y, and z, respectively, and do not state X, Y, and Z as long as no ambiguity arises.

Let us denote the sample distribution by P. We assume that samples are collected right after the update of X. Let $N(x, y, z)$ be the number of (x, y, z) occurrences in T samples. Then, the sample distribution is

$$P(x, y, z) = N(x, y, z)/T.$$

The other quantities such as $N(y)$ or $P(y)$ are also defined similarly.

For HG and complete HG, the conditional distribution $P(x|y)$ converges to the corresponding $\pi(x|y) = \pi(x|z)$. However, in the case of general DHG, $\pi(x|z)$ can differ from the one of the other $\pi(x|z')$ in the same bin, and $\pi(x|z)$ and corresponding $\pi(x|y)$ can be different. Therefore, the convergence of $P(x|y)$ to $\pi(x|y)$ or $\pi(x|z)$ does not necessarily hold. In fact, we can expect that, the difference between $P(x|y)$ in DHG and the distribution $q(x|y)$ decreases with the increase in T, where $q(x|y)$ is a weighted average of conditional distributions in the bin as follows:

$$q(x|y) = \sum_{z'} \pi(x|z')P(z'|y). \tag{1}$$

It is because the difference

$$\begin{aligned} N(x, y) &- \textstyle\sum_{z'} \pi(x|z')N(y, z') \\ &= N(x, y) - N(y) \textstyle\sum_{z'} \pi(x|z')P(z'|y) \\ &= N(y)P(x|y) - N(y)q(x|y) \end{aligned}$$

Algorithm 6 Update of x_i in the unified formulation of HG-type algorithms

```
Let $Z = \boldsymbol{x}_{-i}$.
Draw $Y$ according to $\pi(Y \mid Z)$.
Draw $X$ and update $w_{i,Y}$ by herding.
$x_i \leftarrow X$.
```

is equal to the difference between the initial value and the final value of the weight $w_{i,y}$, which is bounded. For HG, it holds that $q(x|y) = \pi(x|z)$ where z is the only state that corresponds to y, and $P(x|y)$ converges to $q(x|y)$.

4.2 Difference between the target distribution and sample distribution

Here, we discuss the difference between π and P. By the definitions of the variables and MRF, the target distribution is $\pi(x, y, z) = \pi(z)\pi(x|z)\pi(y|z)$, and the difference of distributions P and π is

$$\begin{aligned} &P(x, y, z) - \pi(x, y, z) \\ &\quad = P(x, y, z) - P(z)\pi(x|z)\pi(y|z) \\ &\qquad + [P(z) - \pi(z)]\pi(x|z)\pi(y|z). \end{aligned} \tag{2}$$

In HG-type algorithms, X is drawn according to Y, and the information of Z is not used after the draw of Y. Thus, $P(x|y)$ is easier to evaluate than $P(x|y, z)$. It is useful to think that X and Z are independent given Y and compare $P(x|y)P(z|y)$, instead of $P(x, z|y)$, with $\pi(x|y)\pi(z|y)$. We evaluate the difference of $P(x, z|y)$ and $P(x|y)P(z|y)$ in another term. Thus, the first two terms of (2) are decomposed as

$$\begin{aligned} &P(x, y, z) - P(z)\pi(x|z)\pi(y|z) \\ &\quad = P(y)[P(x, z|y) - P(x|y)P(z|y)] \\ &\qquad + [P(y)P(x|y)P(z|y) - P(z)\pi(x|z)\pi(y|z)]. \end{aligned} \tag{3}$$

In the second term of (3), the difference of the conditional distribution of x between the samples $P(x|y)$ and the target $\pi(x|z)$ can be evaluated using the convergence of $P(x|y)$ to $q(x|y)$ and the difference between $q(x|y)$ and $\pi(x|z)$ as follows:

$$\begin{aligned} &P(y)P(x|y)P(z|y) - P(z)\pi(x|z)\pi(y|z) \\ &\quad = P(y, z)[P(x|y) - q(x|y)] \\ &\qquad + P(z)[q(x|y)P(y|z) - \pi(x|z)\pi(y|z)] \\ &\quad = P(y)[P(x|y) - q(x|y)]P(z|y) \\ &\qquad + P(z)[P(y|z) - \pi(y|z)]q(x|y) \\ &\qquad + P(z)[q(x|y) - \pi(x|z)]\pi(y|z). \end{aligned} \tag{4}$$

4.3 Decomposition of the numerical integration error

As a practical measure of estimation accuracy, we evaluate estimation error. Let $f(x, z)$ be the target function and

$$D = \sum\nolimits_{x,y,z} [P(x, y, z) - \pi(x, y, z)] f(x, z)$$

be the estimation error. By using Eqs. (2), (3), and (4), the estimation error can be decomposed as

$$\begin{aligned} D = &\sum\nolimits_{x,y,z} P(y)[P(x, z|y) - P(x|y)P(z|y)] f(x, z) \\ &+ \sum\nolimits_{x,y,z} P(y)[P(x|y) - q(x|y)]P(z|y) f(x, z) \\ &+ \sum\nolimits_{x,y,z} P(z)[P(y|z) - \pi(y|z)]q(x|y) f(x, z) \\ &+ \sum\nolimits_{x,y,z} P(z)[q(x|y) - \pi(x|z)]\pi(y|z) f(x, z) \\ &+ \sum\nolimits_{x,y,z} [P(z) - \pi(z)]\pi(y|z)\pi(x|z) f(x, z). \end{aligned} \tag{5}$$

From this equation, a loose bound of $|D|$ is obtained by using $\|f\| = \max_{x,z} |f(x, z)|$ as a coefficient. However, we can make a tighter bound that considers the characteristics of the target function.

Let f be written as a sum of a function of X, that of Z, a constant, and a residual:

$$f(x, z) = f^X(x) + f^Z(z) + \check{f}(x, z) + C.$$

The contribution of X and Z to the target function f is represented by f^X and f^Z, respectively. Although decomposition is not unique here, we can obtain a tighter bound by making the norm of $\check{f}$ smaller. Several terms in (5) become zero. For example, it holds that

$$\begin{aligned} &\sum\nolimits_{x,y,z} P(y)[P(x|y) - q(x|y)]P(z|y) f^Z(z) \\ &\quad = \sum\nolimits_{y,z} P(y)P(z|y) f^Z(z) \left[\sum\nolimits_x P(x|y) - \sum\nolimits_x q(x|y)\right] \\ &\quad = \sum\nolimits_{y,z} P(y)P(z|y) f^Z(z)(1 - 1) \\ &\quad = 0. \end{aligned}$$

Moreover, if Y is deterministically selected, then $P(y|z) = \pi(y|z)$ holds and the third term in (5) is zero. It becomes positive if we consider a more general case where Y is drawn randomly (see Sect. 6.5), but here we consider the deterministic case.

Therefore, Eq. (5) is reduced to

$$\begin{aligned} D = &\sum\nolimits_{x,y,z} P(y)[P(x, z|y) - P(x|y)P(z|y)]\check{f}(x, z) \\ &+ \sum\nolimits_{x,y,z} P(y)[P(x|y) \end{aligned}$$

$$
\begin{aligned}
&- q(x|y)]P(z|y)[f^X(x) + \check{f}(x,z)] \\
&+ \sum\nolimits_{x,z} P(z) \left[\sum\nolimits_y \pi(y|z) q(x|y) \right.\\
&\left. -\pi(x|z)\right] \left[f^X(x) + \check{f}(x,z) \right] \\
&+ \sum\nolimits_{x,z} [P(z) - \pi(z)] \pi(x|z) f(x,z). \qquad (6)
\end{aligned}
$$

We denote conditional expectation by adding a subscript to the function. For example,

$$\check{f}_{P(Z|Y)}(x,y) = \sum\nolimits_z P(z|y)\check{f}(x,z).$$

Then, we obtain

$$
\begin{aligned}
|D| \le\ &\|\check{f}\| \sum\nolimits_y P(y) \sum\nolimits_{x,z} |P(x,z|y) - P(x|y)P(z|y)| \\
&+ (\|\check{f}_{P(Z|Y)}\| + \|f^X\|) \\
&\times \sum\nolimits_y P(y) \sum\nolimits_x |P(x|y) - q(x|y)| \\
&+ (\|\check{f}\| + \|f^X\|) \\
&\times \sum\nolimits_z P(z) \sum\nolimits_x \left| \sum\nolimits_y q(x|y)\pi(y|z) - \pi(x|z) \right| \\
&+ \|f_{\pi(X|Z)}\| \sum\nolimits_z |P(z) - \pi(z)|, \qquad (7)
\end{aligned}
$$

where the norm is the maximum norm. Let us denote factors by

$$
\begin{aligned}
D^{\text{cor}} &= \sum\nolimits_y P(y) \sum\nolimits_{x,z} |P(x,z|y) - P(x|y)P(z|y)|, \\
D^{\text{herding}} &= \sum\nolimits_y P(y) \sum\nolimits_x |P(x|y) - q(x|y)|, \\
D^{\text{approx}} &= \sum\nolimits_z P(z) \sum\nolimits_x \left| \sum\nolimits_y q(x|y)\pi(y|z) - \pi(x|z) \right|, \\
D^z &= \sum\nolimits_z |P(z) - \pi(z)|,
\end{aligned}
$$

and

$$
\begin{aligned}
\lambda_{\text{cor}} &= \|\check{f}\|, \\
\lambda_{\text{herding}} &= \|\check{f}_{P(Z|Y)}\| + \|f^X\|, \\
\lambda_{\text{approx}} &= \|\check{f}\| + \|f^X\|, \\
\lambda_z &= \|f_{\pi(X|Z)}\|,
\end{aligned}
$$

and we obtain the following bound:

$$
\begin{aligned}
|D| \le\ &\lambda_{\text{cor}} D^{\text{cor}} + \lambda_{\text{herding}} D^{\text{herding}} + \lambda_{\text{approx}} D^{\text{approx}} \\
&+ \lambda_z D^z \qquad (8)
\end{aligned}
$$

Thus, we obtained four components of estimation error. We will interpret and evaluate each component in detail in the following sections, and here we give a brief description for each component. Here, D^{cor} represents the correlation of X and Z given Y. As we described in Sect. 3.2, they are correlated in the sample distribution and D^{cor} represents the asymptotic bias arising from the correlation. We use D^{herding} to represent the convergence of the distribution of X given Y. This will be shown after to decrease at the rate of $O(1/T)$ because of the property of herding. D^{approx} represents the difference of $\pi(x|z)$ and the corresponding $q(x|y)$. In fact, $D^{\text{approx}} = 0$ for HG and can be positive for DHG, as will be shown later. Thus, D^{approx} represents the approximation error of DHG. D^z represents the convergence of the distribution of Z.

Numerical Example A numerical example of the behavior of the error components is shown in Fig. 3. We used a Boltzmann machine with $N = 8$ variables which take values in $\{-1, +1\}$ and are connected in numerical order in a ring. The model parameters, the coupling constants W_{ij} and biases B_i, are drawn uniformly random from $[0, 1]$ or $[-1, 1]$. We used the original HG; that is, no dummy edges are added and it corresponds to a Herding(2) in Fig. 1. We updated variables in numerical order, and collected samples right after each update of x_N. We collected $T = 2^{24}$ samples for a single measurement and averaged the results of 10 measurements. We calculated D^z, D^{herding}, D^{approx}, and D^{cor} from the collected samples. We also calculated $D^{\text{all}} = \sum_{x,z} |P(x,z) - \pi(x,z)|$ as the measure of the estimation error of the joint distribution. The true distribution π was directly calculated according to the definition of the Boltzmann machine. The other experiments in the remainder of this paper are conducted using the same procedure.

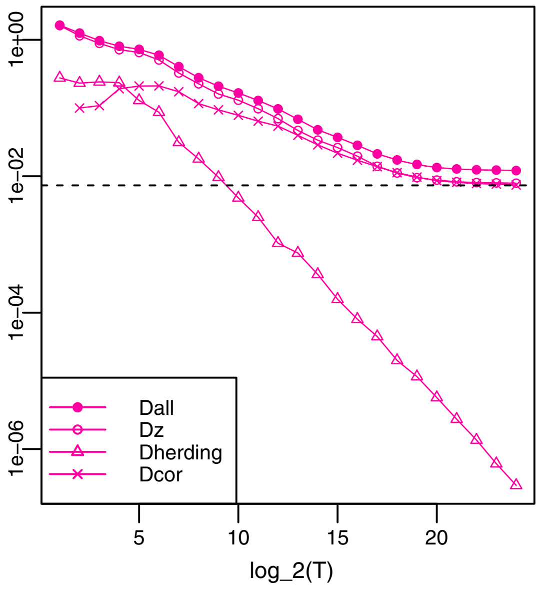
We can see that D^{cor} reaches some asymptotic value, and D^{all} and D^z behave similarly to each other. The dotted line shows the estimation of the asymptotic value of D^{cor} in Sect. 5.2.

In particular, D^{herding} decreases faster than other quantities, with an order of $O(1/T)$. Here, D^{approx} is omitted from the plot because it is zero for HG (see Sect. 6.2).

Figure 4 shows the numerical example of the effect of the selection of B in DHG. We can see that D^{herding} becomes small and D^{approx} and D^{cor} become large by decreasing B. The results are consistent with the effect of weight sharing as shown in Fig. 2.

5 Asymptotic error caused by weight sharing

In the following, we consider a case of a Boltzmann machine with 0–1 variables, and we evaluate the error components of (8). We denote the evaluated value by putting a tilde on it (e.g., $\widetilde{D}^{\text{cor}}$). First, we focus on D^{cor}.



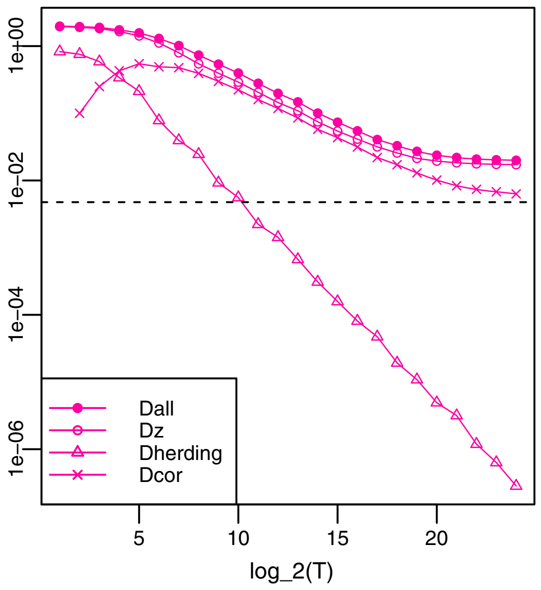


Fig. 3 Convergence measure behaviors of HG for a Boltzmann machine on a ring with $N = 8$ variables which take values in $\{-1, +1\}$ and random parameter configuration in (top) $[0, 1]$ and (bottom) $[-1, 1]$. The dotted line shows the estimated asymptotic value of D^{cor} in Sect. 5.2

5.1 Error from the sample correlation: D^{cor}

D^{cor} represents the difference between $P(x|y)P(z|y)$ and $P(x, z|y)$. If X and Z are independent given Y, this must be zero. However, X and Z are correlated in a sample distribution as we discussed in Sect. 3.2. In fact, the sequences of X and Z have temporal correlations, and they cause the correlation of X and Z. Here, we investigate the temporal correlation of each and evaluate the asymptotic value of D^{cor} as $T \to \infty$.

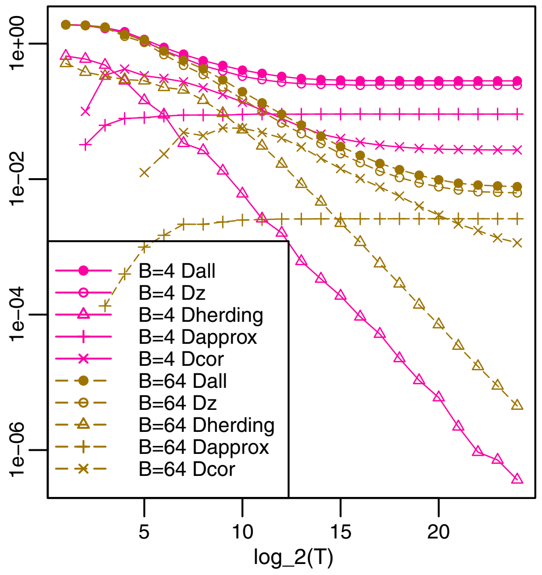


Fig. 4 Convergence measure behaviors of DHG. The target model is a Boltzmann machine on a fully connected graph with $N = 8$ variables which take values in $\{-1, +1\}$ and random parameter configuration in $[-0.5, 0.5]$

First, we investigate the temporal correlation of X. We focus on one value y of Y and let $t = \pi(X = 1|Y = y)$. We can assume $t < 1/2$ without loss of generality. Right after drawing $X = 1$, the value of w after the update is in $[-1 + t, -1 + 2t]$. Then, we must draw $X = 0$ in the next step because $-1 + 2t < 0$. Similarly, in the next step after drawing $X = 0$, the value of w is in $[-1 + 2t, t]$ and the probability of drawing $X = 1$ is greater than t. Although we can narrow the range of possible w values by considering more previous samples, we simply use only the last sample and assume w is distributed equally in the range $[-1+t, -1+2t]$ or $[-1 + 2t, t]$. Using this idea, the sample sequence of X can be approximated by a Markov chain and we can obtain a transition probability from x' to x. We denote this by $T_y(x' \to x)$. Specifically, the transition matrix is given by

$$\begin{pmatrix} T_y(0 \to 0) & T_y(1 \to 0) \\ T_y(0 \to 1) & T_y(1 \to 1) \end{pmatrix} = \begin{pmatrix} \frac{1-2t}{1-t} & 1 \\ \frac{t}{1-t} & 0 \end{pmatrix}. \tag{9}$$

Next, we investigate the temporal correlation of Z. Let us assume here that each variable of Z is updated following the original Gibbs sampling procedure. For all y, let $T_y((x', z') \to z)$ be the transition probability that a chain beginning from $(X, Y, Z) = (x', y, z')$ comes to $(Y, Z) = (y, z)$ in the next time such that X is updated with $w_{i,y}$. In other words, we split the chain of HG with occurrences of $Y = y$, and each fragment is aggregated into one Markov step denoted by T_y.

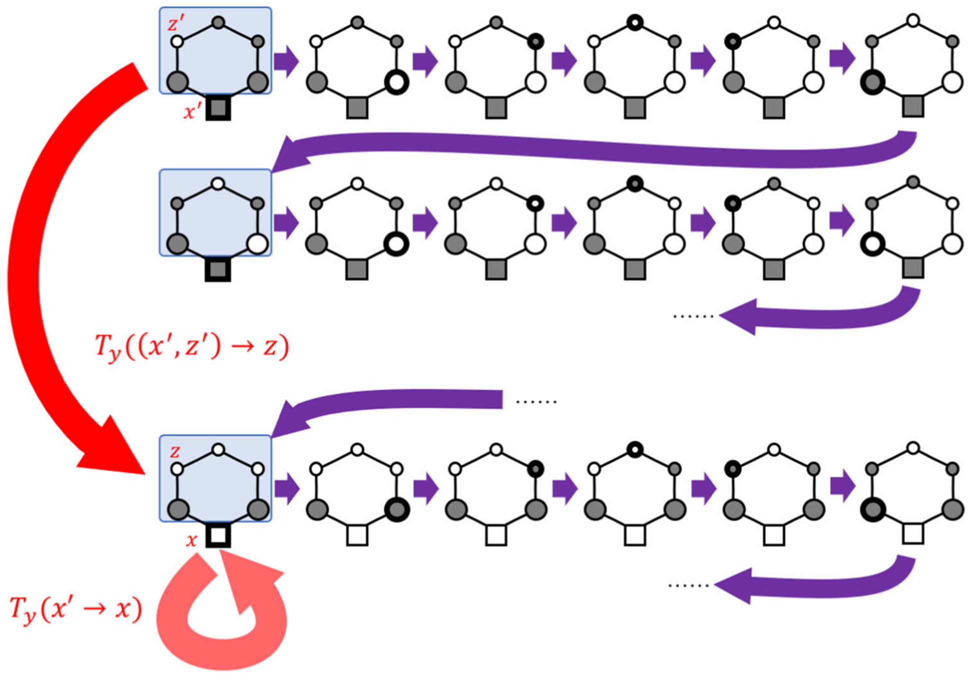


Fig. 5 Example of aggregating Markov chain transitions into T_y

Figure 5 shows an example illustrating how T_y is defined. The circles and squares denote binary variables that take the state of 0 (white) or 1 (gray). Each hexagon consists of six variables denoting a sample. The square in the bottom of the hexagon denotes X, and the five circles above denote Z. Two circles neighboring the square determine Y. Let us assume that $Y = y$ when both of these circles are gray. The circles and squares drawn by thick lines denote the variables to be updated. The sampling proceeds along the arrows, and the variables are updated in a cyclic order.

In the first state of the top row, it satisfies $Y = y$. This sample is expressed by (x', y, z'). It does not satisfy $Y = y$ in the first state of the second row, when X is updated, and it satisfies $Y = y$ in the bottom row for the first time after (x', y, z'). All the transitions from (x', y, z') to this state $(*, y, z)$ are aggregated together into $T_y((x', z') \to z)$, which is denoted by the red arrow. Right after this joined transition, X is updated following the transition $T_y(x' \to x)$.

The distribution to which the sample distribution converges can be approximated by the following distribution:

$$\begin{aligned}&\sum_{x',z'} \pi(x', z'|y)T_y(x' \to x)T_y((x', z') \to z)\\ &\quad= \sum_{x'} \pi(x'|y)T_y(x' \to x)\sum_{z'} \pi(z'|x', y)T_y((x', z') \to z)\\ &\quad= \sum_{x'} \pi(x'|y)T_y(x' \to x)T_y(x' \to z),\end{aligned}$$

where $T_y(x' \to z)$ is a probability distribution on Z defined as

$$T_y(x' \to z) = \sum_{z'} \pi(z'|x', y)T_y((x', z') \to z).$$

In other words, $T_y(x' \to z)$ represents the dependency of the distribution of Z on the previous value of X. For simplicity of notation, we denote the mixture of $T_y(0 \to z)$ and $T_y(1 \to z)$ as

$$T_y(s \to z) = (1 - s) \cdot T_y(0 \to z) + s \cdot T_y(1 \to z) \quad (\forall s \in [0, 1]).$$

Then, we obtain an approximation of asymptotic distribution of P as

$$\begin{aligned}P(z|y) &\to \sum_{x'} \pi(x'|y)T_y(x' \to z) = T_y(t \to z),\\ P(z|x, y) &= \frac{P(x, z|y)}{P(x|y)}\\ &\to \frac{\sum_{x'} \pi(x'|y)T_y(x' \to x)T_y(x' \to z)}{\pi(x|y)}.\end{aligned}$$

By substituting (9), we obtain

$$\begin{aligned}P(z|0, y) &\to \frac{(1 - 2t) \cdot T_y(0 \to z) + t \cdot T_y(1 \to z)}{1 - t}\\ &= T_y\left(\frac{t}{1 - t} \to z\right),\\ P(z|1, y) &\to \frac{t \cdot T_y(0 \to z) + 0 \cdot T_y(1 \to z)}{t}\\ &= T_y(0 \to z).\end{aligned}$$

We can evaluate the contribution for D^{cor} of each y to a value proportional to t^2 as follows:

$$\begin{aligned}
&\sum\nolimits_{x,z} |P(x,z|y) - P(x|y)P(z|y)| \\
&\quad = \sum\nolimits_{x,z} P(x|y)|P(z|x,y) - P(z|y)| \\
&\quad \to (1-t)\sum\nolimits_{z} \left| T_y(\frac{t}{1-t} \to z) - T_y(t \to z) \right| \\
&\qquad + t\sum\nolimits_{z} |T_y(0 \to z) - T_y(t \to z)| \\
&\quad = 2t^2 \sum\nolimits_{z} \left| T_y(0 \to z) - T_y(1 \to z) \right| \\
&\quad \equiv \widetilde{D}_y^{\text{cor}},
\end{aligned}$$

Finally, the evaluation of D^{cor} is obtained as

$$\widetilde{D}^{\text{cor}} \equiv \sum\nolimits_{y} P(y)\widetilde{D}_y^{\text{cor}}.$$

Summarizing the above, we can say that the amount of D^{cor} is affected by both the distribution of X given Y, which is represented by t, and the dependency of the distribution of Z on the previous value of X, which is represented by $\left|T_y(0 \to z) - T_y(1 \to z)\right|$.

5.2 Difference between $T_y(0 \to z)$ and $T_y(1 \to z)$

We show a numerical example of the difference between $T_y(0 \to z)$ and $T_y(1 \to z)$. Using Boltzmann machines in the numerical example in Sect. 4, we collected $T = 2^{32}$ samples by normal Gibbs sampling and calculated the measure of the difference,

$$D_y^{depend} \equiv \sum_z |T_y(0 \to z) - T_y(1 \to z)|$$

which is used for the evaluation of $\widetilde{D}_y^{\text{cor}}$. Here, t is calculated by $t = \min_x \pi(x|y)$. By using it, we also calculated the estimation of the contributions to D^{cor} of each y by

$$\pi(y)\widehat{D}_y^{\text{cor}} \equiv \pi(y) \times 2t^2 D_y^{depend}.$$

Table 1 shows the result. Each row shows the calculated quantity for each $y = (x_1, x_7)$ value. The fifth row shows the numerical estimation of the asymptotic values of D^{cor} that are calculated by $\sum_y \pi(y)\widehat{D}_y^{\text{cor}}$. The estimation is also shown by the dashed lines in Fig. 3. We can see that we have obtained good evaluations of the asymptotic values.

5.3 Relationship between D^{cor} and B

The setting of B, the number of weights per variable in DHG, affects the asymptotic value of D^{cor}. A small bin width with large B decreases the probability of each $w_{i,y}$ being used and increases the length of the interval between updates of the same $w_{i,y}$. Then, each fragment of the chain that is aggregated into $T_y(x' \to z)$ includes more Gibbs steps. Because the chain mixes better if it is longer, we can expect the difference between $T_y(0 \to z)$ and $T_y(1 \to z)$ to decrease, and so does D^{cor}.

We also numerically investigated the relationship between B and the asymptotic value of D^{cor}. We used the same Boltzmann machine as in Fig. 4. From Fig. 6, we can see the asymptotic value of D^{cor} shows a roughly exponential decay as B increases.

Table 1 Differences between $T_y(0 \to z)$ and $T_y(1 \to z)$ and the estimation of D^{cor} for model used in Fig. 3

y	$\pi(y)$	t	D_y^{depend}	$\pi(y)\widehat{D}_y^{\text{cor}}$
(0, 0)	0.0235	0.0642	1.49e−01	2.88e−05
(1, 0)	0.0935	0.4849	1.51e−01	6.64e−03
(0, 1)	0.0472	0.4485	4.71e−03	8.93e−05
(1, 1)	0.8358	0.0559	1.12e−01	5.84e−04
				7.34e−03
(0, 0)	0.3633	0.0723	7.67e−02	2.91e−04
(1, 0)	0.4109	0.2218	2.65e−02	1.07e−03
(0, 1)	0.0803	0.3451	2.15e−02	4.11e−04
(1, 1)	0.1455	0.3417	8.76e−02	2.98e−03
				4.75e−03

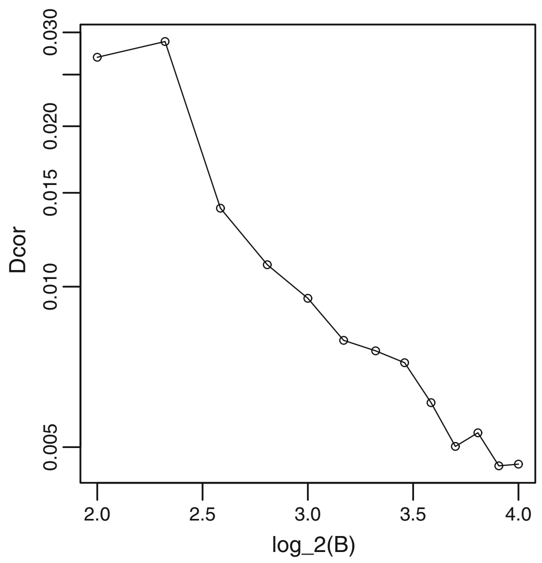


Fig. 6 Decrease in the asymptotic value of D^{cor} in terms of the increase in B. The target model is the same as that in Fig. 4. For all cases in this plot, D^{cor} ceases decreasing before $T = 2^{24}$, and we record the value of D^{cor} in $T = 2^{24}$ as the asymptotic value

5.4 Bounded-error Gibbs sampling for reducing D^{cor}

We have seen that the temporal correlation of X causes an asymptotic bias. In this section, we propose a way to reduce the temporal correlation and the asymptotic value of D^{cor}.

Algorithm 2 shows the herding procedure for a Bernoulli distribution, and Algorithm 3 shows the herding procedure for a multinomial distribution. If we set the initial value of the weight variable to $w_x^{(0)} = 0$, it holds that

$$\delta_x \equiv |p - \sum x^{(t)}/T| = |w_x^{(T)} - w_x^{(0)}| = |w_x^{(T)}|,$$

and the herding procedure becomes a greedy minimization of $\max_x \delta_x = \max_x |w_x|$. The fast convergence of herding is due to the boundedness of $|\delta_x|$. Therefore, we can ease the minimization condition to merely a bounding condition. Algorithms 7 and 8 show the proposed procedure. In this algorithm, w_x is updated in the same way as the herding, but the deterministic update of x by maximization is done only when $\max_x w_x$ exceeds the given threshold c. In the other cases, the ordinary random sampling is used. Setting c appropriately, a sufficiently large proportion of samples is drawn by independent random samplings so that the temporal correlation decreases. In addition, still bounded δ_x keeps the fast convergence property of herding, while the upper bound of the error increases. It becomes herding when $c = 0$ and it coincides with an ordinary random sampling in the limit of $c \to \infty$. We call this procedure bounded-error sampling and the HG-type algorithm using this procedure bounded-error Gibbs (BEG).

Algorithm 7 Bounded-error sampling for a Bernoulli distribution

```
if $|w| > c$ then
    $x^{(t)} \leftarrow \mathbb{I}(w > 0)$.
else
    $x^{(t)} \sim Ber(p)$.
end if
$w \leftarrow w + p - x^{(t)}$.
```

Algorithm 8 Bounded-error sampling for a multinomial distribution

```
if $\max_x w_x > c$ then
    $x^{(t)} \leftarrow \mathrm{argmax}_x w_x$.
else
    $x^{(t)} \sim Multi(p_x)$.
end if
$w_x \leftarrow w + p_x - \mathbb{I}(x^{(t)} = x)$, for all $x$.
```

Numerical Experiment We numerically compare BEG with Gibbs sampling and DHG. BEG with $c = 0$ and $c = \infty$ coincide with DHG and Gibbs sampling, respectively. In addition

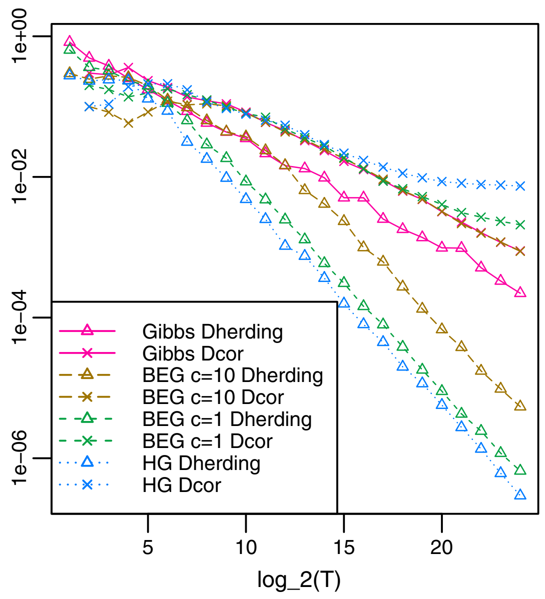


Fig. 7 Comparison of BEG with Gibbs sampling and HG. The target model is a Boltzmann machine, which is same as the model for the top panel of Fig. 3

to $c = 0, \infty$, we ran BEG with $c = 1, 10$. Figure 7 shows the result. We can see that larger c decreases D^{cor}, while it increases D^{herding}.

If we take too large c, $\max_x w_x$ will not exceed the threshold c during the sampling and the algorithm becomes essentially the same as Gibbs sampling. We can see this situation in $\log_2(T) \le 12$ of D^{herding} for BEG with $c = 10$ in Fig. 7. Therefore, we have to take sufficiently small c compared to the number of times each bin is used, if it can be estimated beforehand. Although small c leads to the convergence speed-up, too small c leads to the problem of the bias. Whether it is too small or not is dependent on T, the number of samples to be collected. In practice, we can use the following preliminary experiment to choose the best c. Run BEG and Gibbs sampling simultaneously for desired time period T, and record the decrease in the difference of the obtained distributions (or estimations calculated). If the period T is small and the bias is insignificant, the difference will continue to decrease throughout. If the decrease stops, it means the bias becomes significant there and the chosen c is turned out to be too small. Then, we have to increase c.

6 HG-type numerical integration

In this section, we focus on the other error components, and the estimation error, especially in the terms D^{herding} and D^{approx}, which represent the behavior of the conditional distribution of x.

6.1 Error from herding: D^{herding}

$$
\begin{aligned}
P(x|y) - q(x|y) &= \frac{TP(x,y) - TP(y)q(x|y)}{TP(y)} \\
&= \frac{N(x,y) - N(y)q(x|y)}{TP(y)}.
\end{aligned}
$$

Let $\delta_{x|y} = N(x,y) - N(y)q(x|y)$. This equals to the difference between the first value and the last value of $w_{i,y}$. We can rewrite D^{herding} as follows:

$$
\begin{aligned}
D^{\text{herding}} &= \sum\nolimits_y P(y) \sum\nolimits_x |P(x|y) - q(x|y)| \\
&= \frac{1}{T} \sum\nolimits_{x,y} |\delta_{x|y}|.
\end{aligned}
$$

By assuming that $|\delta_{x|y}|$ is bounded above by a constant c, we obtain the evaluation as

$$
D^{\text{herding}} \leq \frac{1}{T} \sum\nolimits_{x,y} c = \frac{2cB}{T} \equiv \widetilde{D}^{\text{herding}}. \tag{10}
$$

Here, D^{herding} decreases at the rate of $O(1/T)$ as well as herding, so we can say it represents the effect of herding in HG-type algorithm. In DHG, D^{herding} increases by taking more small bins with larger B. This leads to the extension of the Gibbs-like period shown in Fig. 2.

6.2 Error from approximation of herding dynamics: D^{approx}

D^{approx} represents the difference between $\sum_y q(x|y)\pi(y|z)$ and $\pi(x|z)$. For HG, it holds that $q(x|y) = \pi(x|z)$ for the corresponding y and z (i.e., $\pi(y|z) = 1$), by the definition (1). Then, D^{approx} is equal to zero for HG. We can say that D^{approx} represents the estimation bias of DHG that is caused by the approximation of $\pi(x|z)$ for multiple z in the same bin by the representative $q(x|y)$.

As in Sect. 2.4, we assume that bins are determined by $0 = \theta_0 < \theta_1 < \cdots < \theta_{B-1} < \theta_B = 1$ and have the same width of $1/B$. By the definition (1),

$$
q(1|b) = \sum\nolimits_{z'} \pi(1|z')P(z'|b).
$$

If $\pi(z'|b) > 0$, z' is in the bth bin and $\pi(1|z') \in (\theta_b, \theta_{b+1}]$. Thus, the weighted average $q(1|b)$ is also in $(\theta_b, \theta_{b+1}]$. Therefore, if z is included in the bth bin, then it holds that

$$
\begin{aligned}
&\pi(1|z) \in [\theta_b, \theta_{b+1}], \\
&\sum\nolimits_y q(1|y)\pi(y|z) = q(1|b) \in [\theta_b, \theta_{b+1}],
\end{aligned}
$$

and

$$
\begin{aligned}
&\pi(0|z) \in [1-\theta_{b+1}, 1-\theta_b], \\
&\sum\nolimits_y q(0|y)\pi(y|z) = q(0|b) \in [1-\theta_{b+1}, 1-\theta_b].
\end{aligned}
$$

Thus, it holds that

$$
\left| \sum\nolimits_y q(x|y)\pi(y|z) - \pi(x|z) \right| \leq \max\nolimits_b(\theta_{b+1} - \theta_b) = 1/B,
$$

and D^{approx} is bounded as

$$
\begin{aligned}
D^{\text{approx}} &= \sum\nolimits_z P(z) \sum\nolimits_x \left| \sum\nolimits_y q(x|y)\pi(y|z) - \pi(x|z) \right| \\
&\leq \sum\nolimits_x P(z)(1/B) \\
&= 2/B \equiv \widetilde{D}^{\text{approx}}.
\end{aligned} \tag{11}
$$

That is, the amount of D^{approx} is proportional to the width of bins.

6.3 Convergence of Z: D^z

We use D^z to represent the difference between the target distribution and the sample distribution for all variables except X. However, all variables and sampling procedures affect the dynamics of weight variables and it is very complicated to analyze accurately. However, as well as the conclusions from Sect. 5.1, we believe they at least converge similarly to conventional Gibbs sampling, except for the asymptotic bias. Note that D^z does not necessarily converge to zero because the distribution of each variable included in Z also has the asymptotic bias.

6.4 Effect of the integrand function and bias–variance trade-off

We can also discuss the effect of the target function f on the performance of the HG-type algorithm using the bound (8). If f is a univariate function of X, we can make $f^Z(z) = \check{f}(x,z) = 0$. Because $\lambda_{\text{cor}} = \|\check{f}\| = 0$, the contribution of D^{cor} disappears. In the general case, f is also a function of z, so that $\|\check{f}\| > 0$. Therefore, an asymptotic bias arises from D^{cor}. Here, D^z is affected by all other variables than X and has a positive asymptotic value. Even in the case of the univariate target function $f(x,z) = f^X(x)$, $\lambda_z = \|f_{\pi(x|z)}\|$ does not necessarily equal zero and then there can be an asymptotic estimation bias.

We have seen that there is a trade-off between the initial performance and the asymptotic bias as schematized in Fig. 2. By using the proposed formulation, we can express this trade-off in a more explicit way and obtain a rough estimation of the optimal weight sharing. Let us put aside the effect of D^{cor} and D^z and express the trade-off as minimizing $\lambda_{\text{herding}}\widetilde{D}^{\text{herding}} + \lambda_{\text{approx}}\widetilde{D}^{\text{approx}}$. If we assume that bins have equal width $1/B$, by using the results (10) and (11), it can be written more simply;

$$\lambda_{\text{herding}}\widetilde{D}^{\text{herding}} + \lambda_{\text{approx}}\widetilde{D}^{\text{approx}} = \lambda_{\text{herding}}\frac{2cB}{T} + \lambda_{\text{approx}}\frac{2}{B}$$

and the bound is minimized by setting

$$B \simeq \sqrt{\frac{T\lambda_{\text{approx}}}{c\lambda_{\text{herding}}}} = \sqrt{\frac{T(\|\check{f}\| + \|f^X\|)}{c(\|\check{f}_{P(Z|Y)}\| + \|f^X\|)}},$$

where we note that it is merely a rough estimate of optimal B.

6.5 Randomly discretized herded Gibbs for reducing D^{approx}

By introducing randomness, we can reduce the asymptotic bias from D^{approx}.

Algorithm 9 Updating x_i in randomly discretized herded Gibbs

```
Let $p = \pi(x_i = 1|\boldsymbol{x}_{(i)})$.
Find an integer $b$ such that $\theta_b < p \le \theta_{b+1}$.
Let $r = (\theta_{b+1} - p)/(\theta_{b+1} - \theta_b)$.
Set $Y = b$ with probability $r$, or $Y = b + 1$ otherwise.
$x_i \leftarrow \mathbb{I}(w_{i,Y} > 0)$.
$w_{i,Y} \leftarrow w_{i,Y} + \theta_Y - x_i$.
```

Algorithm 9 shows the proposed algorithm, and we call it randomly discretized herded Gibbs (RDHG). First, we take fixed probability values $0 = \theta_0 < \theta_1 < \cdots < \theta_{B-1} < \theta_B = 1$ as in DHG, and use weight variables $w_{i,b}$ generating a herding sample sequence for a fixed probability θ_b, where $b \in \{0, \ldots, B\}$. Thus, we set $q(x_i = 1|Y = b) = \theta_b$. The conditional distribution $\pi(x|z)$ is a Bernoulli distribution $Ber(p)$ for some p. It can be regarded as a mixture of two distributions $Ber(\theta_b)$ and $Ber(\theta_{b+1})$ for such b that $p \in [\theta_b, \theta_{b+1}]$. Then, it holds that $p = r\theta_b + (1 - r)\theta_{b+1}$, where $r = (\theta_{b+1} - p)/(\theta_{b+1} - \theta_b)$. We can make $\sum_y q(x|y)\pi(y|z) = \pi(x|z)$ by randomly drawing Y according to the probability r.

Then, although the first, second, and fifth term of (5) are the same as DHG, the fourth term becomes zero, in other words $D^{\text{approx}} = 0$. However, the third term of (7) becomes nonzero and then a new error component appears:

$$\left|\sum_{x,y,z}\pi(z)[P(y|z) - \pi(y|z)]q(x|y)f(x,z)\right|$$
$$= \left|\sum_{x,y,z}\pi(z)[P(y|z) - \pi(y|z)]q(x|y)[\check{f}(x,z) + f^X(x)]\right|$$
$$\le (\|\check{f}_{q(X|Y)}\| + \|f^X_{q(X|Y)}\|)\sum_z \pi(z)\sum_y |P(y|z) - \pi(y|z)|.$$

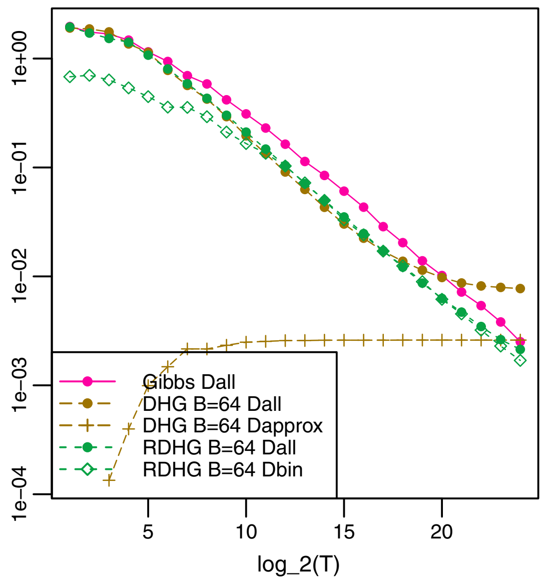


Fig. 8 Comparison of RDHG with Gibbs sampling and DHG. The target model is the same as that in Fig. 4

Let us denote the new component by

$$D^{\text{bin}} = \sum_z \pi(z)\sum_y |P(y|z) - \pi(y|z)|$$
$$\lambda_{bin} = \|\check{f}_{q(X|Y)}\| + \|f^X_{q(X|Y)}\|,$$

and the error bound for RDHG is

$$|D| \le \lambda_{\text{cor}}D^{\text{cor}} + \lambda_{\text{herding}}D^{\text{herding}} + \lambda_{\text{bin}}D^{\text{bin}} + \lambda_z D^z.$$

Here, D^{bin} converges to zero in $O(T^{-1/2})$.

Numerical Experiment Figure 8 shows the performance of RDHG compared with those of Gibbs sampling and DHG. $D^{all} = \sum_{x,z}|P(x,z) - \pi(x,z)|$ is the same as those in the previous experiments shown in Figs. 3 and 4. We show only error components that differ among the algorithms in the plot. Here, D^z is omitted because it behaves similarly to D^{all}. A fully connected $N = 8$ graph whose parameters are drawn uniformly random in $[-0.5, 0.5]$ is used. We can see DHG reaches the larger asymptotic value. We can also see that DHG and RDHG outperform Gibbs sampling in D^{all} within an interval of T, and that of RDHG is longer than DHG.

6.6 Comparison with the Rao–Blackwell estimator

Consider the case of estimating the marginal distribution of X. In other words, the target function is

$$f(x,z) = \mathbb{I}(x = a).$$

Then, we can set

$$f^X(x) = \mathbb{I}(x = a)$$
$$f^Z(z) = \check{f}(x, z) = 0.$$

Thus, the bound (8) is calculated as

$$\begin{aligned}|D| &\leq \|\check{f}\|D^{\text{cor}} + (\|\check{f}_{P(Z|Y)}\| + \|f^X\|)D^{\text{herding}} \\ &\quad + (\|\check{f}\| + \|f^X\|)D^{\text{approx}} + \|f_{\pi(X|Z)}\|D^z \\ &= \|f^X\|D^{\text{herding}} + \|f^X\|D^{\text{approx}} + \|f_{\pi(X|Z)}\|D^z \\ &\leq \frac{1}{2}D^{\text{herding}} + \frac{1}{2}D^{\text{approx}} \\ &\quad + \frac{1}{2}(\max_z \pi(x = a|z) - \min_z \pi(x = a|z))D^z.\end{aligned}$$

Several terms related to $\|\check{f}\|$ have disappeared, and this suggests that HG is effective in estimating marginal distributions of single variables.

It is also effective for marginalizing the target, especially for estimating the marginal distribution of X. That is, X is integrated out and the new target $f_{\pi(X|Z)}(z) = \sum_x \pi(x|z)f(x, z)$ instead of $f(x, z)$ is used. This estimator is also called the Rao–Blackwell (RB) estimator. We can think of this as eliminating the error of the conditional distribution $P(X|Z)$. In the context of our analysis, this means that D^{herding} vanishes, whereas in HG we can only reduce D^{herding} by introducing herding. In particular, the error of the RB estimator is

$$\begin{aligned}|D_{RB}| &= \left|\sum_z [P(z) - \pi(z)] \sum_x \pi(x|z)f(x, z)\right| \\ &= \left|\sum_z [P(z) - \pi(z)] f_{\pi(x|z)}(z)\right| \\ &\leq \|f_{\pi(x|z)}\| \sum_z |P(z) - \pi(z)|. \end{aligned} \tag{12}$$

In comparison with (7), which is the estimation error of HG, $|D_{RB}|$ includes the last term of (7), which correspond to D^z, and does not include the term corresponding to D^{herding}. In addition, it also does not include bias terms such as D^{cor} and D^{approx}. Thus, the RB estimator appears to be better than HG, or at least equal to it.

Still, the HG-type algorithm has another advantageous point over the RB estimator. Assume that the RB target $f_{\pi(X|Z)}$ is written as a sum of univariate functions and residuals as

$$f_{\pi(X|Z)}(z) = \sum_{j \in V\setminus\{i\}} f^j(x_j) + f^R(z).$$

Here, f^j represents the contribution of x_j that is included in Z to the RB target $f_{\pi(X|Z)}$. Similar to the decomposition of f in Sect. 4.3, we can obtain a tight bound of $|D|$ for HG if $\|f^R\|$ in the decomposition is small.

The contribution of D^z to the estimation error $|D|$ of HG, which is the same as $|D_{RB}|$, is written as follows:

$$\begin{aligned}&\left|\sum_z [P(z) - \pi(z)] f_{\pi(x|z)}(z)\right| \\ &\leq \sum_j \sum_{x_j} |P(x_j) - \pi(x_j)| f^j(x_j) \\ &\quad + \sum_z |P(z) - \pi(z)| f^R(z). \end{aligned} \tag{13}$$

The first term decreases quickly because of the effectiveness of HG for estimating the marginal distribution for each single variable as explained before. In other words, we can say that the distribution of samples generated by HG-type algorithm is automatically "Rao–Blackwellized" for all variables.

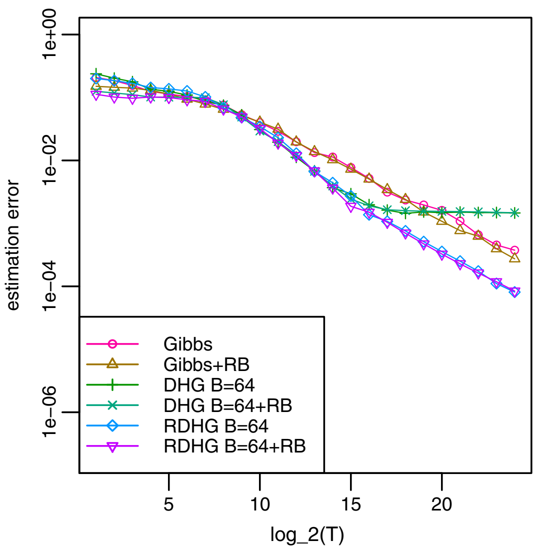
Numerical Experiment We conducted an experiment of estimating $E[x_8]$ in a fully connected Boltzmann machine with $N = 8$ variables which take values in $\{-1, +1\}$. We used two graphs whose parameters are drawn uniformly from $[0, 0.5]$ and $[0, 0.1]$, respectively. As in the previous experiments, the true expectation is calculated directly. We compare the original Gibbs sampling, that with RB improvement, and DHG and RDHG with $B = 64$. Figure 9 shows the results. For all graphs, RDHG outperforms Gibbs. Furthermore, it also outperforms the Gibbs with the RB improvement only.

If the parameters of Boltzmann machine are small, $f_{\pi(X|Z)}$ can be well approximated by a linear function. Then, we expect that the error becomes small because we can make the norm of f^R in (13) small. In the bottom panel of Fig. 9, the improvement in DHG or RDHG over RB is greater than that of the top panel, which is consistent with the discussion.

According to the result, applying RB to DHG or RDHG may further improve the performance. This result can be explained as follows. The estimation error is mainly divided into two parts, the error corresponding to D^{herding} and D^z. If the parameter values of Boltzmann machine are small, the error corresponding to D^z is small and the large portion of the error is determined by D^{herding}. While D^{herding} decreases fast because of the HG, it cannot be zero within a finite sample size. Therefore, in this case RB makes the further improvement on the DHG and RDHG.

7 Estimation with finite samples in machine learning

In introduction, we stated that we often have a limited time relative to the high dimension of the variables in machine learning problems and it is worth reducing the estimation error as quickly as possible even at the expense of its asymptotic consistency. In this section, we show numerical experiments on the task of image reconstruction as an



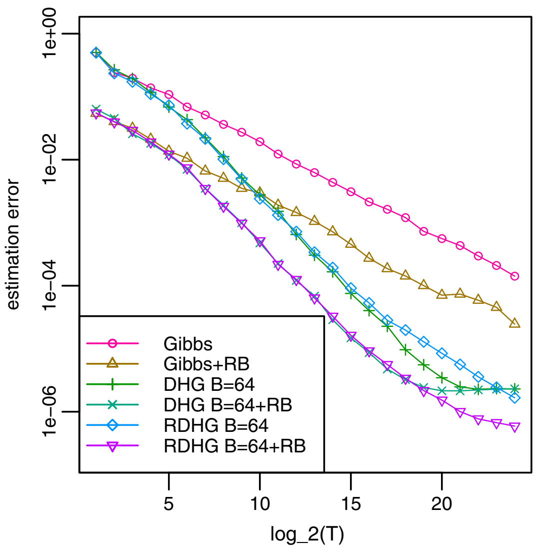


Fig. 9 Estimation error of $E[x_8]$ in an $N = 8$ fully connected Boltzmann machine. The absolute errors of estimation for 100 times are averaged. The parameters are drawn from (top) [0, 0.5] and (bottom) [0, 0.1]

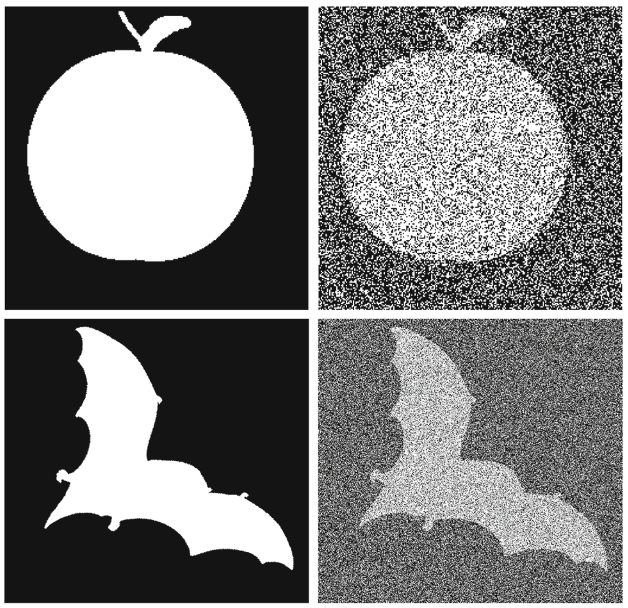


Fig. 10 Left: the samples of MPEG-7 CE Shape-1 Part-B data set. Right: the samples of noisy images generated from the data set

example of this situation, and show that HG-type algorithms perform well.

Image reconstruction is a well-known application of MRFs. We consider two-valued images each of whose pixels takes a value in $x_i \in \{-1, +1\}$. The pixels neighboring each other are naturally expected to have the same value with a high probability. We can write down this as a probabilistic model as

$$p(\mathbf{x}) \propto \exp\left(\sum_{(i,j)\in E} J_{ij} x_i x_j\right),$$

where E is a set of neighboring variables. We set $J_{ij} = 1$ in this experiment. We assume that the image was degraded with the random flips of pixels, and we reconstruct the original image from the observed noisy image. Let $\mathbf{y}$ denote the observed pixels. The probability obtaining them is

$$p(\mathbf{y} \mid \mathbf{x}) = \prod_i p(y_i \mid x_i) \propto \exp\left(\sum_i b x_i y_i\right),$$

where $b = \frac{1}{2}\log\left(\frac{1-p}{p}\right)$. Thus, the posterior distribution is described as the following MRF:

$$p(\mathbf{x}|\mathbf{y}) \propto \exp\left(\sum_{(i,j)\in E} J_{ij} x_i x_j + \sum_i b y_i x_i\right)$$

We use MCMC to collect T samples from this MRF and reconstruct the pixels with the sample average and the threshold 0.5.

We use MPEG-7 CE Shape-1 Part-B data set (Jan Latecki et al. 2000). It is a set of two-valued images representing 70 classes of shapes, and the length of a side varies from 38 to 1116. We generated noisy images by flipping pixels independently and randomly with the probability of $p = 0.3$. Figure 10 shows two samples of the data set and the noisy images generated from them. We randomly take 100 images from the data set and conduct the image reconstruction for them. We initialize MCMC by setting each pixel uniformly randomly and collect $T = 31$ samples. This corresponds to the situation where we have limited number of samples relative to the dimension of variable. We calculate the proportion

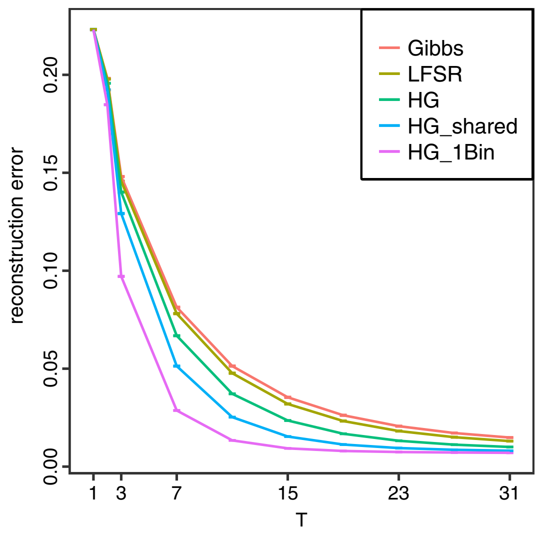


Fig. 11 The proportion of incorrectly reconstructed pixels with respect to the number of the samples collected. The mean of 10 trials is shown, and the standard deviation, which is very small, is shown by the error bar

of incorrectly reconstructed pixels to all pixels and report the mean and the standard deviation of 10 trials.

We use three HG-type algorithms in addition to the ordinary Gibbs sampling. The first (HG) is the HG whose variables have $2^4 = 16$ weight variables for each. The second (HG-shared) is the HG with weight sharing, and each variable has five weight variables. We can naturally classify the condition of neighboring variables into five bins by the sum of them. The third (HG-1Bin) is the HG obtained by applying weight sharing to the maximum. Namely, we put all the conditions into one bin and then each variable uses only one weight variable.

In addition to the HG-type algorithms, we also applied QMCMC. We use the sequence generated by LFSR following the procedure of Tribble (2007) as the driving sequence of Gibbs sampling. We use $1 + x^3 + x^5$ as the feedback polynomial and $g = 3$ as the offset.

We show the result of each algorithm in Fig. 11. All HG-type algorithms outperform the ordinary Gibbs sampling, and HG-1Bin shows the best performance. This is consistent with our view shown in Sect. 3.3 and Fig. 2; namely, the estimation error decreases with the increase in the weight sharing degree when the number of samples is small. It is also worth noting that HG-shared and HG-1Bin yield error reduction within only a few samples.

We have the threshold 0.5 for reconstruction, and the excessive accuracy is not needed if the averages fall into the right side. This is considered as a reason why the drawback of asymptotic bias was limited.

In Fig. 11, QMCMC outperforms the Gibbs sampling, but HG-type algorithms perform better, especially for small T.

These results illustrate pros and cons of HG-type algorithms and QMCMC. Basically, both types of the algorithms are based on equidistributed number sequences generated by deterministic algorithms. However, a large difference is in the auto-correlation. HG-type algorithms actively exploit the negative auto-correlation in samples to reduce the error, especially for a small sample size, while it leads to a positive asymptotic value of D^{cor} as a side effect. In QMCMC, the auto-correlation is eliminated for the sake of consistency, and thus we lose the error reduction for a small sample size, as shown in Fig. 11. In addition, the driving vectors in QMCMC are designed to be equally distributed in an N-dimensional space, where N is the number of variables. This means that equidistributedness becomes significant after sufficiently large number of samples are generated so that the driving vectors are well distributed in the N-dimensional space. Therefore, the variance reduction in the QMCMC becomes effective for a large number of samples.

8 Conclusion

In this study, we investigated the convergence behavior of HG-type algorithms by introducing the notion of weight sharing. We investigated the effects of weight sharing and showed HG-type algorithm's superiority to conventional MCMC in relation to it. Weight sharing improves the convergence rate at the initial period of an iteration, while it causes the asymptotic bias in estimation. The effect is determined by the degree of weight sharing. If the size of the generated sample sequence is not so large, the bias is not significant and HG can estimate more accurately than Gibbs sampling.

To make the analysis more systematic, we decomposed the error of estimation into four components and interpreted them as the effects of herding and weight sharing. One of the components represents the effect of the herding procedure for the focused variable (D^{herding}), which accelerates the convergence, and another represents the convergence of the other variables (D^{z}). The other two components are related to the asymptotic bias. One is the approximation error caused by weight sharing among the conditioning states which have different conditional distributions (D^{approx}). Another error component is the effect of the temporal correlation of samples caused by the herding procedure (D^{cor}).

Inspired by the analysis, we proposed two improvement techniques to reduce the biases. BEG introduces randomness for the herding procedure to reduce the temporal correlation of samples and the resulting bias. RDHG introduces randomness for selecting the weight variable and reduces the bias from the weight sharing in DHG.

By using the analysis, we demonstrated that with selection of the algorithm and its parameter values, we can control the convergence behavior. There is some trade-off between the algorithm and parameter selection, and we have to appropriately select them according to the problem to be applied. We also showed that the efficiency of the estimation by using HG-type algorithms is affected by the target function, which largely depends on the problem. We should investigate the characteristics of target functions in real problems in the future, to widely reveal the practicality of HG-type algorithms. The experiment of image reconstruction in this paper is an example for it. Although we focused on a case of a Boltzmann machine with binary variables, analysis for MRFs with discrete variables can be performed in a similar manner.

The analysis proposed in this paper includes many approximations, and a mathematically rigorous guarantee is still needed. It will be a difficult task as suggested in Chen et al. (2016), but our study can be a first step toward completing this task.

MCMC algorithms, including Gibbs sampling, generally suffer from a problem of bad mixing. That should be also true for HG-type algorithms. In our analysis, we did not consider the effect of mixing explicitly, and it is hidden in the dynamics of z. Technically, if the chain mixes badly, the convergence of D^z dominates the behavior of $|D|$ and the effect of herding becomes insignificant. Many works exist for improving the mixing of MCMC, and we need to integrate them into HG-type algorithms.

Acknowledgements This work is supported in part by ImPACT Program of Council for Science, Technology and Innovation (Cabinet Office, Government of Japan) and by a Grant-in-Aid for JSPS Fellows (17J02174) and Scientific Research (16KT0105).

References

Chen, Y., Welling, M., Smola, A.: Super-samples from kernel herding. In: Proceedings of the Twenty-Sixth Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI), pp. 109–116 (2010)

Chen, S., Dick, J., Owen, A.B.: Consistency of Markov chain quasi-Monte Carlo on continuous state spaces. Ann. Stat. **39**(2), 673–701 (2011)

Chen, Y., Bornn, L., de Freitas, N., Eskelin, M., Fang, J., Welling, M.: Herded Gibbs sampling. J. Mach. Learn. Res. **17**(10), 1–29 (2016)

Chentsov, N.N.: Pseudorandom numbers for modelling Markov chains. USSR Comput. Math. Math. Phys. **7**(3), 218–33 (1967)

Dick, J., Rudolf, D., Zhu, H.: Discrepancy bounds for uniformly ergodic Markov chain quasi-Monte Carlo. Ann. Appl. Probab. **26**(5), 3178–3205 (2016)

Eskelinen, M.: Herded Gibbs and discretized herded Gibbs sampling. Master's thesis, The University of British Columbia (2013)

Jan Latecki, L., Lakämper, R., Eckhard, U.: Shape descriptors for non-rigid shapes with a single closed contour (2000)

Tribble, S. D.: Markov chain Monte Carlo algorithms using completely uniformly distributed driving sequences. PhD thesis, Stanford University (2007)

Welling, M.: Herding dynamic weights for partially observed random field models. In: Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence (UAI), pp. 599–606 (2009a)

Welling, M.: Herding dynamical weights to learn. In: Proceedings of the 26th Annual International Conference on Machine Learning (ICML), pp. 1121–1128 (2009b)

Welling, M., Chen, Y.: Statistical inference using weak chaos and infinite memory. J. Phys.: Conf. Ser. **233**(1), 012005 (2010)