Interdisciplinary Information Sciences Vol. 18, No. 1 (2012) 55–63

ISSN 1340-9050 print/1347-6157 online
DOI 10.4036/iis.2012.55

Learning Algorithm for Boltzmann Machines Using Max-Product Algorithm and Pseudo-Likelihood

Muneki YASUDA*, Junya TANNAI and Kazuyuki TANAKA

Graduate School of Information Sciences, Tohoku University, Sendai 980-8579, Japan

Received December 12, 2011; final version accepted October 30, 2012

Boltzmann machines are parametric probabilistic models for the statistical machine learning, forming Markov random fields. Owing to their normalization constant, inference and learning in Boltzmann machines are generally classified under NP-hard problems. Maximum pseudo-likelihood estimation is an effective approximate learning method for Boltzmann machines. However, in principle, we cannot use this method for incomplete data sets, except for some special cases. In this paper, we propose a new learning algorithm for Boltzmann machines with incomplete data sets by generating a pseudo-complete data set from a given incomplete data using the max-product algorithm and the Markov chain Monte Carlo method, and then, by applying maximum pseudo-likelihood estimation to the pseudo-complete data set.

KEYWORDS: deep learning, Boltzmann machine, EM algorithm, max-product belief propagation, pseudo-likelihood estimation

1. Introduction

Recently, statistical machine learning schemes based on the maximum likelihood framework have been widely adapted in information sciences [1, 2]. The object of statistical machine learning is to find a generalization underlying an observed real-world data set, that is, to find a suitable generative model of the observed data set.

Boltzmann machines are parametric probabilistic models for statistical machine learning [3]. They consist of visible and hidden nodes (sometimes called latent nodes) with undirected symmetrical connections between nodes. In Boltzmann machines, the states of all nodes are determined stochastically, and the probability distributions of the states of all nodes can be regarded as a Markov random field (MRF) at their equilibrium state [4]. Owing to their high flexibility and rich structure, Boltzmann machines can be potentially employed in various applications. Indeed, several types of Boltzmann machines such as restricted Boltzmann machines (RBMs) [5] and deep belief networks (DBNs) [6] have been successfully employed in various applications. Unfortunately, most Boltzmann machines are intractable and often suffer from NP-hard problems. Hence, some approximate methods for learning and inference in Boltzmann machines have been investigated by many researchers.

Maximum pseudo-likelihood estimation (MPLE) [7] is an approximate learning method for MRFs, and it is employed for the learning of fully visible Boltzmann machines, which are Boltzmann machines with no hidden nodes [8]. Using MPLE, one can obtain learning solutions within a polynomial computational cost. Furthermore, the solutions are equivalent to or better than those of the contrastive divergence learning, which is one of the most familiar techniques in Boltzmann machine learning [5]. However, in principle, MPLE cannot be directly applied to Boltzmann machines with hidden nodes because the MPLE method requires a complete date set, except for some special cases such as RBMs.

In this paper, by combining MPLE with the max-product algorithm (MPA) [1, 2] (also known as the max-sum algorithm), we propose a new learning algorithm for Boltzmann machines with hidden nodes. The MPA is a belief propagation algorithm that gives maximum a posteriori (MAP) configurations of MRFs having tree structures. We create a pseudo-complete data set from a given incomplete data set by interpolating unobserved data using MPA and the Markov chain Monte Carlo (MCMC) method in order to apply MPLE to Boltzmann machines with hidden nodes. Although the proposed learning method requires sparseness among hidden nodes owing to the MPA, there are no other structural limitations. We verify the validity of our learning method via our numerical simulations. The remainder of this paper is organized as follows. In §2, we introduce a Boltzmann machine and an expectation-maximization (EM) algorithm for it. In §3, we propose a new learning algorithm for the Boltzmann machine introduced in §2. In §4, we describe numerical simulations of the proposed method, and we compare our method with the learning method proposed by Welling and Hinton [9]. Finally in §5, we provide some concluding remarks.

* Corresponding author. E-mail: muneki@smapip.is.tohoku.ac.jp

2. Boltzmann Machine

Let us consider an undirected graph $G(\Omega, L)$ consisting of n nodes and some links; each node is labeled as i. Let $\Omega = \{1, 2, \ldots, n\}$ be the set of all nodes. A link between nodes i and j is labeled as (i, j). Let L be the set of all links. Because we consider undirected links, (i, j) and (j, i) denote the same link. We assign a discrete random variable x_i to node i. We divide Ω into two different sets; one is the set of visible nodes, V, and the other is the set of hidden nodes, H. Visible nodes are input-output nodes, and hidden nodes (sometimes called latent nodes) control the degree of freedom of a system. We define the probabilistic model for the graph as

$$P_\Omega(\boldsymbol{x}_V, \boldsymbol{x}_H \mid \boldsymbol{\theta}, \boldsymbol{w}) := \frac{1}{Z_\Omega(\boldsymbol{\theta}, \boldsymbol{w})} \exp(-E_\Omega(\boldsymbol{x}_V, \boldsymbol{x}_H; \boldsymbol{\theta}, \boldsymbol{w})), \tag{2.1}$$

$$E_\Omega(\boldsymbol{x}_V, \boldsymbol{x}_H; \boldsymbol{\theta}, \boldsymbol{w}) := -\sum_{i \in \Omega} \theta_i x_i - \sum_{(i,j) \in L} w_{ij} x_i x_j, \tag{2.2}$$

where $Z_\Omega(\boldsymbol{\theta}, \boldsymbol{w})$ is the partition function

$$Z_\Omega(\boldsymbol{\theta}, \boldsymbol{w}) := \sum_{\boldsymbol{x}_\Omega} \exp(-E_\Omega(\boldsymbol{x}_V, \boldsymbol{x}_H; \boldsymbol{\theta}, \boldsymbol{w})).$$

Here, we use the notation $\boldsymbol{x}_A := \{x_i \mid i \in A \subseteq \Omega\}$. The model parameters $\boldsymbol{\theta}$ and $\boldsymbol{w}$ play the roles of biases and couplings, respectively. The probabilistic model in eq. (2.1) represents a Boltzmann machine [3, 4].

For learning in the Boltzmann machine, we determine the values of the model parameters by using a given set of data. Note that the set of data is incomplete; we have no data for hidden nodes $\boldsymbol{x}_H$. We only have data for visible nodes $\boldsymbol{x}_V$. Given a set of data, the Boltzmann machine is usually trained using maximum likelihood estimation (MLE). In MLE, given M data values $\mathcal{D} := \{d_i^{(\mu)} \mid i \in V, \mu = 1, 2, \ldots, M\}$, one maximizes the (log-) likelihood function

$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{w}) := \sum_{\boldsymbol{x}_V} Q_\mathcal{D}(\boldsymbol{x}_V) \ln P_V(\boldsymbol{x}_V \mid \boldsymbol{\theta}, \boldsymbol{w}) \tag{2.3}$$

with respect to the model parameters, where

$$Q_\mathcal{D}(\boldsymbol{x}_V) := \frac{1}{M} \sum_{\mu=1}^{M} \prod_{i \in V} \delta(x_i, d_i^{(\mu)}), \quad \delta(x, y) := \begin{cases} 1 & x = y \\ 0 & x \neq y \end{cases}$$

is the empirical distribution of the given data set and

$$P_V(\boldsymbol{x}_V \mid \boldsymbol{\theta}, \boldsymbol{w}) := \sum_{\boldsymbol{x}_H} P_\Omega(\boldsymbol{x}_V, \boldsymbol{x}_H \mid \boldsymbol{\theta}, \boldsymbol{w})$$

is the marginal distribution of the Boltzmann machine. Owing to the existence of hidden nodes, the maximization of the likelihood function in eq. (2.3) can be archived by using the EM algorithm. In the EM algorithm, one obtains the desired model parameters by the following procedure. In the E-step, one estimates the Q-function defined as

$$\mathcal{Q}(\boldsymbol{\theta}, \boldsymbol{w} \mid \boldsymbol{\theta}^*, \boldsymbol{w}^*) := \sum_{\boldsymbol{x}_\Omega} P_{H|V}(\boldsymbol{x}_H \mid \boldsymbol{x}_V, \boldsymbol{\theta}^*, \boldsymbol{w}^*) Q_\mathcal{D}(\boldsymbol{x}_V) \ln P_\Omega(\boldsymbol{x}_V, \boldsymbol{x}_H \mid \boldsymbol{\theta}, \boldsymbol{w}), \tag{2.4}$$

where

$$P_{H|V}(\boldsymbol{x}_H \mid \boldsymbol{x}_V, \boldsymbol{\theta}, \boldsymbol{w}) := \frac{P_\Omega(\boldsymbol{x}_V, \boldsymbol{x}_H \mid \boldsymbol{\theta}, \boldsymbol{w})}{P_V(\boldsymbol{x}_V \mid \boldsymbol{\theta}, \boldsymbol{w})}.$$

In the M-step, one maximizes the Q-function with respect to $\boldsymbol{\theta}$ and $\boldsymbol{w}$,

$$\{\hat{\boldsymbol{\theta}}, \hat{\boldsymbol{w}}\} = \arg\max_{\{\boldsymbol{\theta}, \boldsymbol{w}\}} \mathcal{Q}(\boldsymbol{\theta}, \boldsymbol{w} \mid \boldsymbol{\theta}^*, \boldsymbol{w}^*),$$

and subsequently, one sets $\{\boldsymbol{\theta}^*, \boldsymbol{w}^*\} \leftarrow \{\hat{\boldsymbol{\theta}}, \hat{\boldsymbol{w}}\}$ and returns to the E-step. This procedure is repeated until convergence. However, because the computational cost of the M-step increases exponentially with n, except for some special cases, one cannot execute the M-step for large n.

3. Maximum Pseudo-Likelihood Learning Using Pseudo-Complete Data Set

To prevent a computational complexity of $O(e^n)$ in the EM algorithm, we extend MPLE and apply it to the EM algorithm. MPLE is known as a tractable approximate learning algorithm for fully visible Boltzmann machines, having a computational complexity of $O(|L|)$ [8]. MPLE provides a unique solution that is equivalent to a solution obtained using MLE in the case where there are no model error and no statistical error [7, 8]. However, MPLE requires a complete data set. Hence, it cannot be directly applied to the EM algorithm because the given date set is incomplete.

3.1 Approximate Q-function with pseudo-complete data set

In this section, we propose a new method to execute the EM algorithm introduced in the previous section by using MPLE with a pseudo-complete data generated from the given incomplete data. The basic concept is as follows. For the μth observed data $\boldsymbol{d}^{(\mu)} := \{d_i^{(\mu)} \mid i \in V\}$, and given $\boldsymbol{\theta}^*$, $\boldsymbol{w}^*$, we can sample $\boldsymbol{x}_H$ from $P_{H|V}(\boldsymbol{x}_H \mid \boldsymbol{d}^{(\mu)}, \boldsymbol{\theta}^*, \boldsymbol{w}^*)$ in eq. (2.4) by using the MCMC method. Suppose that we get a set of K sampled points from $P_{H|V}(\boldsymbol{x}_H \mid \boldsymbol{d}^{(\mu)}, \boldsymbol{\theta}^*, \boldsymbol{w}^*)$, and we denote kth sampled point by $\boldsymbol{e}^{(k|\mu)}(\boldsymbol{\theta}^*, \boldsymbol{w}^*) := \{e_i^{(k|\mu)}(\boldsymbol{\theta}^*, \boldsymbol{w}^*) \mid i \in H\}$,

$$\{\boldsymbol{e}^{(k|\mu)}(\boldsymbol{\theta}^*, \boldsymbol{w}^*) \mid k = 1, 2, \ldots, K\} \underset{K \text{ samples}}{\sim} P_{H|V}(\boldsymbol{x}_H \mid \boldsymbol{d}^{(\mu)}, \boldsymbol{\theta}^*, \boldsymbol{w}^*).$$

Thus, for one observed data value, we have K sampled points for hidden nodes. If we interpreted the K sampled data points as pseudo-observations for hidden nodes, from $\mathcal{D}$ and $\{\boldsymbol{e}^{(k|\mu)}(\boldsymbol{\theta}^*, \boldsymbol{w}^*) \mid k = 1, 2, \ldots, K\}$, we can construct the pseudo-complete data set, for instance, as $\mathcal{C} := \{c_i^{(l)} \mid i \in \Omega, l = 1, 2, \ldots, KM\}$, where

$$c_i^{(l)} := \begin{cases} d_i^{(m)} & i \in V \\ e_i^{(r|m)}(\boldsymbol{\theta}^*, \boldsymbol{w}^*) & i \in H \end{cases},$$

where $m = \lfloor (K + l - 1)/K \rfloor$ and $r = l - (m - 1)K$, where $\lfloor x \rfloor$ is the floor function which gives the largest integer less than or equal to x, i.e., $\lfloor x \rfloor = \max\{n \in \mathbb{Z} \mid n \leq x\}$. The pseudo-empirical distribution for the pseudo-complete data set is constructed as

$$Q_{\mathcal{C}}(\boldsymbol{x}_\Omega \mid \boldsymbol{\theta}^*, \boldsymbol{w}^*) := \frac{1}{KM} \sum_{l=1}^{KM} \prod_{i \in \Omega} \delta(x_i, c_i^{(l)}). \tag{3.1}$$

Using the pseudo-empirical distribution we approximate the Q-function as

$$\mathcal{Q}(\boldsymbol{\theta}, \boldsymbol{w} \mid \boldsymbol{\theta}^*, \boldsymbol{w}^*) \approx \sum_{\boldsymbol{x}_\Omega} Q_{\mathcal{C}}(\boldsymbol{x}_\Omega \mid \boldsymbol{\theta}^*, \boldsymbol{w}^*) \ln P_\Omega(\boldsymbol{x}_V, \boldsymbol{x}_H \mid \boldsymbol{\theta}, \boldsymbol{w}). \tag{3.2}$$

If $K \to \infty$, the above approximation becomes an equality because

$$P_{H|V}(\boldsymbol{x}_H \mid \boldsymbol{x}_V, \boldsymbol{\theta}^*, \boldsymbol{w}^*) Q_{\mathcal{D}}(\boldsymbol{x}_V) = \lim_{K \to \infty} Q_{\mathcal{C}}(\boldsymbol{x}_\Omega \mid \boldsymbol{\theta}^*, \boldsymbol{w}^*).$$

Since eq. (3.2) can be interpreted as the log-likelihood function with a complete data set, we can apply MPLE to the Q-function as

$$\mathcal{Q}(\boldsymbol{\theta}, \boldsymbol{w} \mid \boldsymbol{\theta}^*, \boldsymbol{w}^*) \approx \sum_{\boldsymbol{x}_\Omega} Q_{\mathcal{C}}(\boldsymbol{x}_\Omega \mid \boldsymbol{\theta}^*, \boldsymbol{w}^*) \ln \prod_{i \in \Omega} P_i(x_i \mid \boldsymbol{x}_{-i}, \boldsymbol{\theta}, \boldsymbol{w}) =: \mathcal{Q}_{\mathrm{PL}}(\boldsymbol{\theta}, \boldsymbol{w} \mid \boldsymbol{\theta}^*, \boldsymbol{w}^*), \tag{3.3}$$

where $\boldsymbol{x}_{-i} := \{x_j \mid j \in \Omega \setminus \{i\}\}$ and

$$P_i(x_i \mid \boldsymbol{x}_{-i}, \boldsymbol{\theta}, \boldsymbol{w}) := \frac{P_\Omega(\boldsymbol{x}_V, \boldsymbol{x}_H \mid \boldsymbol{\theta}, \boldsymbol{w})}{\sum_{x_i} P_\Omega(\boldsymbol{x}_V, \boldsymbol{x}_H \mid \boldsymbol{\theta}, \boldsymbol{w})} = \frac{\exp(-E_\Omega(\boldsymbol{x}_V, \boldsymbol{x}_H; \boldsymbol{\theta}, \boldsymbol{w}))}{\sum_{x_i} \exp(-E_\Omega(\boldsymbol{x}_V, \boldsymbol{x}_H; \boldsymbol{\theta}, \boldsymbol{w}))}.$$

The maximization of $\mathcal{Q}_{\mathrm{PL}}(\boldsymbol{\theta}, \boldsymbol{w} \mid \boldsymbol{\theta}^*, \boldsymbol{w}^*)$ is much easier than the maximization of the original Q-function because $\mathcal{Q}_{\mathrm{PL}}(\boldsymbol{\theta}, \boldsymbol{w} \mid \boldsymbol{\theta}^*, \boldsymbol{w}^*)$ is a concave function with respect to $\boldsymbol{\theta}$ and $\boldsymbol{w}$, and its gradients are obtained with a computational complexity of $O(MK|L|)$.

3.2 MCMC method with MPA

If sample points from the distribution $P_{H|V}(\boldsymbol{x}_H \mid \boldsymbol{x}_V, \boldsymbol{\theta}^*, \boldsymbol{w}^*)$ are efficiently obtained, we can construct a practical learning algorithm using the framework presented in §3.1. However, the MCMC method requires a long relaxation procedure before reaching an equilibrium state. The relaxation procedure can potentially become a serious problem because we need to include it in each M-step. If we can start the MCMC method from an appropriate initial configuration, such costly procedures can be avoided.

Let us regard a configuration of $\boldsymbol{x}_H$, which gives the highest probability of $P_{H|V}(\boldsymbol{x}_H \mid \boldsymbol{x}_V, \boldsymbol{\theta}^*, \boldsymbol{w}^*)$ for specific $\boldsymbol{x}_V$, as a sample generated from the MCMC method reaching an equilibrium state, and let us treat this configuration as the initial configuration of the MCMC method. However, in general, finding the configuration is an intractable problem that requires an approximate method. The MPA is a message-passing-type algorithm that gives approximate configurations providing maximum probabilities of Boltzmann machines [1, 2]. Next, we briefly explain the MPA. Let us consider a Boltzmann machine defined on $G(\Omega_0, L_0)$, which takes the form

$$P_{\Omega_0}(\boldsymbol{x}_{\Omega_0}) \propto \left(\prod_{i \in \Omega_0} \phi_i(x_i) \right) \left(\prod_{(i,j) \in L_0} \psi_{ij}(x_i, x_j) \right), \tag{3.4}$$

where $\phi_i(x_i)$ and $\psi_{ij}(x_i, x_j)$ are potential functions for node i and link (i, j), respectively. Note that the Boltzmann machine expressed in eq. (2.1) as well as the distribution $P_{H|V}(\boldsymbol{x}_H \mid \boldsymbol{x}_V, \boldsymbol{\theta}^*, \boldsymbol{w}^*)$ can be represented in the same functional form (see Appendix A). The MPA in the Boltzmann machine is the message-passing-type algorithm,

$$\mathcal{M}_{j\to i}(x_i) = \max_{x_j}\left\{\frac{1}{\mathcal{Z}_{j\to i}}\psi_{ij}(x_i,x_j)\phi_j(x_j)\prod_{k\in\partial_{L_0}(j)\setminus\{i\}}\mathcal{M}_{k\to j}(x_j)\right\}, \tag{3.5}$$

where $\mathcal{Z}_{j\to i}$ is the normalized constant given by

$$\mathcal{Z}_{j\to i} := \sum_{x_i}\psi_{ij}(x_i,x_j)\phi_j(x_j)\prod_{k\in\partial_{L_0}(j)\setminus\{i\}}\mathcal{M}_{k\to j}(x_j).$$

The notation $\partial_{L_0}(i)$ denotes the set of nodes that directly connect to node i through links in L_0, $\partial_{L_0}(i) := \{j \mid (i,j)\in L_0\}$. $\mathcal{M}_{j\to i}(x_i)$ is the message from node j to node i. The messages are defined on all links. We can regard eq. (3.5) as a simultaneous equation with respect to the messages. Using solutions to eq. (3.5), one computes

$$r_i(x_i) \propto \phi_i(x_i)\prod_{j\in\partial_{L_0}(i)}\mathcal{M}_{j\to i}(x_i) \tag{3.6}$$

for all i. Using these quantities, we can approximately obtain the configuration of $\boldsymbol{x}_{\Omega_0}$ which gives the highest probability of $P_{\Omega_0}(\boldsymbol{x}_{\Omega_0})$, $\boldsymbol{x}^*_{\Omega_0} := \arg\max_{\boldsymbol{x}_{\Omega_0}} P_{\Omega_0}(\boldsymbol{x}_{\Omega_0})$, as

$$x_i^* \approx \arg\max_{x_i} r_i(x_i). \tag{3.7}$$

As we can see, the desired configuration is separately determined for each node in the MPA. It is worth noting that if $G(\Omega_0, L_0)$ is a tree, the MPA is exact.

In our framework, we obtain sample points of $\boldsymbol{x}_H$, i.e., $\{\boldsymbol{e}^{(k|\mu)}(\boldsymbol{\theta}^*,\boldsymbol{w}^*) \mid k=1,2,\ldots,K\}$, by the following procedure. For specific $\boldsymbol{d}^{(\mu)}$, $\boldsymbol{\theta}^*$, and $\boldsymbol{w}^*$, we compute an initial configuration of $\boldsymbol{x}_H$, $\boldsymbol{e}^{(0|\mu)}(\boldsymbol{\theta}^*,\boldsymbol{w}^*)$, by using the MPA on the distribution $P_{H|V}(\boldsymbol{x}_H \mid \boldsymbol{d}^{(\mu)},\boldsymbol{\theta}^*,\boldsymbol{w}^*)$. We execute the MCMC method starting from the computed initial configuration, and we get one sample for each of the λ Monte Carlo steps, i.e.,

$$\boldsymbol{e}^{(0|\mu)}(\boldsymbol{\theta}^*,\boldsymbol{w}^*) \underset{\lambda\ \text{steps}}{\longrightarrow} \boldsymbol{e}^{(1|\mu)}(\boldsymbol{\theta}^*,\boldsymbol{w}^*) \underset{\lambda\ \text{steps}}{\longrightarrow} \cdots \underset{\lambda\ \text{steps}}{\longrightarrow} \boldsymbol{e}^{(K|\mu)}(\boldsymbol{\theta}^*,\boldsymbol{w}^*), \tag{3.8}$$

where we let one Monte Carlo step update all the nodes once by using the Gibbs sampler. The above procedure is carried out for all $\mu = 1,2,\ldots,M$.

In the following, we summarize the procedure of the proposed learning method.

PROPOSED LEARNING ALGORITHM

Step 1. Given an incomplete data set $\mathcal{D} = \{\boldsymbol{d}^{(\mu)} \mid \mu = 1,2,\ldots,M\}$, initialize $\boldsymbol{\theta}^*$ and $\boldsymbol{w}^*$.

Step 2. For $\mu = 1,2,\ldots,M$, using the MPA in eqs. (3.5)–(3.7) for $P_{H|V}(\boldsymbol{x}_H \mid \boldsymbol{d}^{(\mu)},\boldsymbol{\theta}^*,\boldsymbol{w}^*)$, generate the initial configuration of $\boldsymbol{x}_H$, i.e., $\boldsymbol{e}^{(0|\mu)}(\boldsymbol{\theta}^*,\boldsymbol{w}^*)$.

Step 3. Using the Gibbs sampler starting from the given initial configuration, generate pseudo-observations for the hidden nodes $\{\boldsymbol{e}^{(k|\mu)}(\boldsymbol{\theta}^*,\boldsymbol{w}^*) \mid k=1,2,\ldots,K, \mu=1,2,\ldots,M\}$ from eq. (3.8).

Step 4. Combining data set $\mathcal{D}$ with the given pseudo-observations for hidden nodes, construct pseudo-complete data set $\mathcal{C}$ and construct the pseudo-empirical distribution in eq. (3.1).

Step 5. Maximize[†] the approximate Q-function in eq. (3.3) with respect to $\boldsymbol{\theta}$ and $\boldsymbol{w}$:

$$\{\hat{\boldsymbol{\theta}},\hat{\boldsymbol{w}}\} \leftarrow \arg\max_{\{\boldsymbol{\theta},\boldsymbol{w}\}} \mathcal{Q}_{\mathrm{PL}}(\boldsymbol{\theta},\boldsymbol{w} \mid \boldsymbol{\theta}^*,\boldsymbol{w}^*).$$

Step 6. Update $\boldsymbol{\theta}^*$ and $\boldsymbol{w}^*$ by

$$\{\boldsymbol{\theta}^*,\boldsymbol{w}^*\} \leftarrow \{\hat{\boldsymbol{\theta}},\hat{\boldsymbol{w}}\}$$

and return to Step 2.

The above procedure is repeated until convergence.

The total computational complexity of this algorithm is $O(MK|L|)$.

The concept of the proposed learning algorithm is quite similar to that of the contrastive divergence learning [5, 10] at the point where non-observed data corresponding to hidden nodes are generated by a sampling method. Our method allows (sparse) links between hidden nodes. In contrast, the ordinary contrastive divergence learning never allow links between hidden nodes. This is the major difference between our method and the ordinary contrastive divergence learning.

4. Experiments

We consider the Boltzmann machine shown in Fig. 1. This Boltzmann machine is an extension of RBMs. Indeed, if one eliminates all the links between nodes in the same layer, this system reduces to an RBM. Although one can apply

[†] To reach convergence, instead of full maximization, it is sufficient to execute a few steps uphill on the pseudo-Q function by a gradient ascent method. In fact, we stop the maximization by 20 steps, with step size being 0.01 in our numerical experiments described in §4.

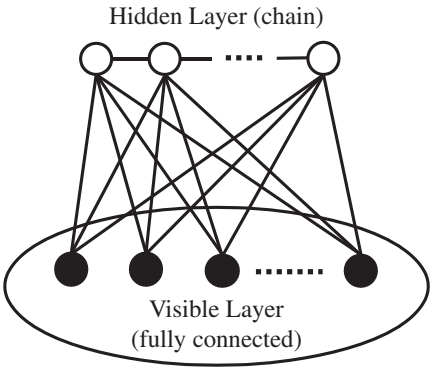

Fig. 1. Boltzmann machine consisting of two different layers, visible and hidden. The closed nodes and the open nodes represent visible nodes and hidden nodes, respectively. Visible nodes are fully connected in the visible layer, and hidden nodes form a chain. Each hidden node connects with all visible nodes.

MPLE to RBMs [11], one cannot directly apply it to this extended system, which has links among hidden nodes, without artifices like those in our method. Let all variables be binary variables, which take values of $+1$ or -1. We let this system be the generative model in our numerical experiments and we generate M data points from this Boltzmann machine. In the generative Boltzmann machine, biases $\boldsymbol{\theta}$ and couplings $\boldsymbol{w}$ are independently drawn from $\mathcal{U}[-h, h]$ and $\mathcal{U}[-J, J]$, respectively, where $\mathcal{U}[a, b]$ is a uniform distribution with interval $[a, b]$.

4.1 A quality of proposed sampling method

In this section, we show a quality of our sampling method using the MPA. In the generative Boltzmann machine defined in the previous section, we set $|V| = 17$, $|H| = 3$, $h = 0.1$, and $J = 0.3$, and then, we generate M data from the generative Boltzmann machine. Although the generated data set includes data points corresponding to hidden nodes, we only see data points corresponding to visible nodes.

Given $\boldsymbol{\theta}$ and $\boldsymbol{w}$, to see how much the pseudo-complete data set, $\mathcal{C}$, created by our sampling method can reconstruct a statistical property of hidden nodes, we compare mean values of $\boldsymbol{x}_H$ evaluated by using the pseudo-empirical distribution with those exactly evaluated using the generative model. Figure 2 shows the mean square errors (MSEs),

$$|H|^{-1} \sum_{i \in H} (\langle x_i \rangle_{Q_{\mathcal{C}}} - \langle x_i \rangle_{\text{exact}})^2,$$

for $M = 100$ and $M = 1000$, against K which is the number of sample points of hidden nodes, where the notation $\langle \cdots \rangle_{Q_{\mathcal{C}}}$ denotes the sample average calculated by using the pseudo-empirical distribution, $Q_{\mathcal{C}}(\boldsymbol{x}_\Omega \mid \boldsymbol{\theta}, \boldsymbol{w})$, and the notation $\langle \cdots \rangle_{\text{exact}}$ denotes the exact average calculated by the generative Boltzmann machine. Note that we know the values of all true parameters in the generative Boltzmann machine in this experiment. There are almost no variations in the MSEs for K. This means that a small K can reconstruct a statistical property of hidden nodes almost as well as a relatively large K. In contrast, the MSEs strongly depend on M. For instance, compare the case of $M = 100$ and $K = 20$ with that of $M = 1000$ and $K = 2$. In the two cases, the total numbers of pseudo-complete data points are the same.

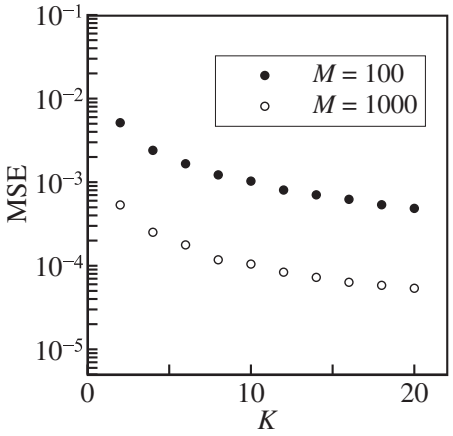

Fig. 2. MSEs of reconstruction by using our sampling method.

The pseudo-complete data set was introduced to evaluate a quantity

$$f^{\dagger}(\boldsymbol{\theta}, \boldsymbol{w}) := \sum_{\boldsymbol{x}_V, \boldsymbol{x}_H} f(\boldsymbol{x}_V, \boldsymbol{x}_H) P_{H|V}(\boldsymbol{x}_H \mid \boldsymbol{x}_V, \boldsymbol{\theta}, \boldsymbol{w}) P_V(\boldsymbol{x}_V \mid \boldsymbol{\theta}, \boldsymbol{w}).$$

for a function $f(\boldsymbol{x}_V, \boldsymbol{x}_H)$. Note that in the numerical experiment in this section, $f(\boldsymbol{x}_V, \boldsymbol{x}_H) = x_i, i \in H$. Given a data point $\boldsymbol{d}^{(\mu)}$, an aim of our sampling method in §3 is to approximate

$$\bar{f}(\boldsymbol{d}^{(\mu)}, \boldsymbol{\theta}, \boldsymbol{w}) := \sum_{\boldsymbol{x}_H} f(\boldsymbol{d}^{(\mu)}, \boldsymbol{x}_H) P_{H|V}(\boldsymbol{x}_H \mid \boldsymbol{d}^{(\mu)}, \boldsymbol{\theta}, \boldsymbol{w})$$

by the Monte Carlo integration:

$$\bar{f}(\boldsymbol{d}^{(\mu)}, \boldsymbol{\theta}, \boldsymbol{w}) \approx \frac{1}{K} \sum_{k=1}^{K} f(\boldsymbol{d}^{(\mu)}, \boldsymbol{e}^{(k|\mu)}(\boldsymbol{\theta}, \boldsymbol{w})) =: \bar{f}_{\mathrm{IS}}(\boldsymbol{d}^{(\mu)}, \boldsymbol{\theta}, \boldsymbol{w}),$$

where $\boldsymbol{e}^{(k|\mu)}(\boldsymbol{\theta}, \boldsymbol{w})$ is the sampled point from $P_{H|V}(\boldsymbol{x}_H \mid \boldsymbol{d}^{(\mu)}, \boldsymbol{\theta}, \boldsymbol{w})$, which is introduced in §3.1. Because the solution to the MPA (approximately) gives the configuration of $\boldsymbol{x}_H$ making $P_{H|V}(\boldsymbol{x}_H \mid \boldsymbol{d}^{(\mu)}, \boldsymbol{\theta}, \boldsymbol{w})$ the highest, the sampled points $\boldsymbol{e}^{(k|\mu)}(\boldsymbol{\theta}, \boldsymbol{w})$ lie around the most important point. This is similar to the importance sampling method. It is thought that this is a reason why a small K gives a similar performance to a relatively large K. Therefore, for a given specific data point, a small K seem to be sufficient.

By the pseudo-complete data set $f^{\dagger}(\boldsymbol{\theta}, \boldsymbol{w})$ is approximated by

$$\begin{aligned} f^{\dagger}(\boldsymbol{\theta}, \boldsymbol{w}) &= \sum_{\boldsymbol{x}_V} \bar{f}(\boldsymbol{x}_V, \boldsymbol{\theta}, \boldsymbol{w}) P_V(\boldsymbol{x}_V \mid \boldsymbol{\theta}, \boldsymbol{w}) \\ &\approx \sum_{\boldsymbol{x}_V} \bar{f}_{\mathrm{IS}}(\boldsymbol{x}_V, \boldsymbol{\theta}, \boldsymbol{w}) Q_{\mathcal{D}}(\boldsymbol{x}_V) = \frac{1}{M} \sum_{\mu=1}^{M} \bar{f}_{\mathrm{IS}}(\boldsymbol{d}^{(\mu)}, \boldsymbol{\theta}, \boldsymbol{w}). \end{aligned}$$

Therefore, even if $\bar{f}_{\mathrm{IS}}(\boldsymbol{d}^{(\mu)}, \boldsymbol{\theta}, \boldsymbol{w})$ is a good approximation of $\bar{f}(\boldsymbol{d}^{(\mu)}, \boldsymbol{\theta}, \boldsymbol{w})$, a poor $Q_{\mathcal{D}}(\boldsymbol{x}_V)$ degrades a total performance. The quality of $Q_{\mathcal{D}}(\boldsymbol{x}_V)$ only depends on M, and each data point for $\boldsymbol{x}_V$ would be generated by a fair sampling. Here, the "fair sampling" means an unbiased sampling differing from the importance sampling. Thus, a small M should be a strong bottleneck of quality of approximation, and therefore, the quality of the pseudo-complete data set is more sensitive for the value of M than that of K.

4.2 Simulation of learning using artificial data

In this section, we show performances of the proposed learning algorithm. We generate $M = 100$ data points from the generative Boltzmann machine defined above, and subsequently, we perform learning with the Boltzmann machine having the same structure as the generative Boltzmann machine. Note that in the learning phase, we only refer to data points corresponding to visible nodes.

We set $|V| = 17$ and $|H| = 3$. Because the system is not very large, we can precisely evaluate the log-likelihood in eq. (2.3). The left plot in Fig. 3 shows the trajectory of the log-likelihood (per node),

$$|V|^{-1} \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{w}), \tag{4.1}$$

evaluated with the parameters obtained by our method, where we set $h = 0.1$, $J = 0.3$, $K = 3$, and $\lambda = 2$. The horizontal axis represents the update step t of our algorithm. The right plot in Fig. 3 shows the mean absolute error of parameters between at step t and at step $t - 1$:

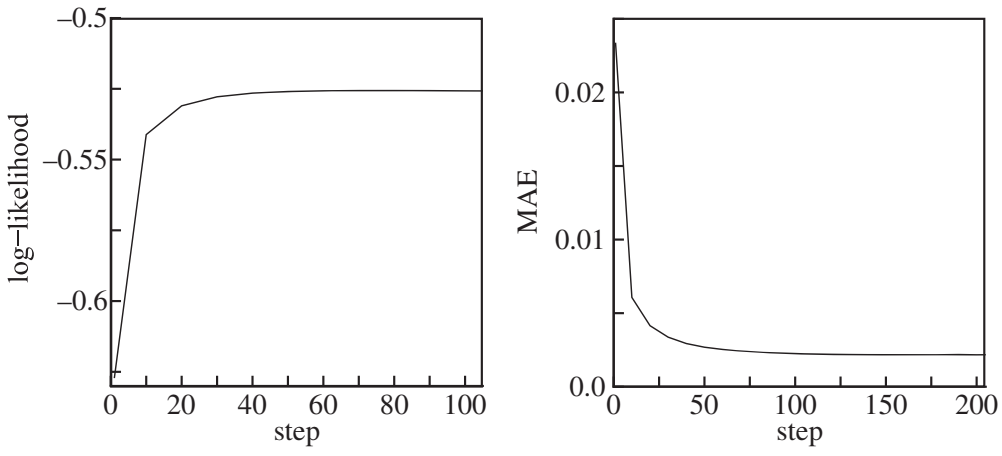

Fig. 3. (Left) Trajectory of the log-likelihood against the update step t. (Right) Trajectory of the mean absolute error of parameters between at step t and at step $t - 1$ shown in eq. (4.2) against the update step t.

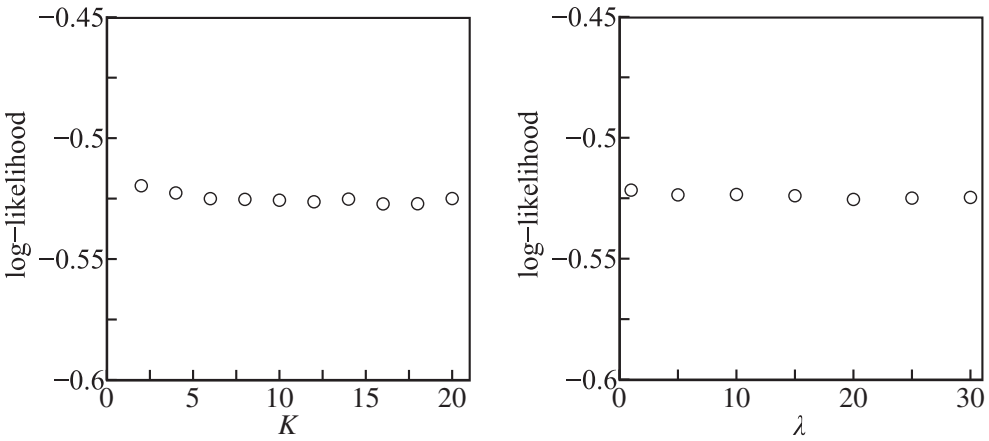

Fig. 4. Dependencies of the log-likelihood on K and λ. On the left, we set $\lambda = 2$, and on the right, we set $K = 4$.

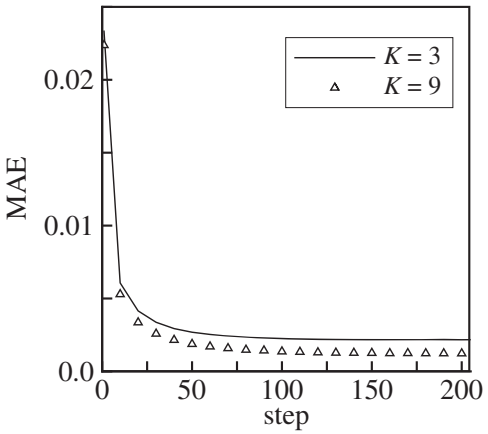

Fig. 5. MAEs in eq. (4.2) for $K = 3$ and 9 against update step t.

$$\mathrm{MAE} := \frac{1}{|\Omega| + |L|}\left(\sum_{i\in\Omega} |\theta_i^{(t)} - \theta_i^{(t-1)}| + \sum_{(i,j)\in L} |w_{ij}^{(t)} - w_{ij}^{(t-1)}|\right). \tag{4.2}$$

We can see that, on average, our algorithm converges by about 100 steps. Figure 4 shows the dependencies of the log-likelihood given by our method on K, which is the number of sample points of hidden nodes, and on λ, which is the number of MCMC steps between two sequential sample points, as in eq. (3.8). The points in these plots represent the values of the log-likelihood at the convergence points. These results indicate that the quality of our method is nearly the same for different values of K and λ. Thus, fortunately, small values of K and λ are sufficient. In particular, the result for K (the left plot of Fig. 4) is consistent with the argument given in §4.1. Although a larger K tends not to change results from the perspective of the likelihood, it can increase the stability of convergence of the algorithm as shown in Fig. 5.

Again, we set $|V| = 17$, $|H| = 3$, and $h = 0.1$. Figure 6 shows the plot of the log-likelihood against J, which controls the correlations among nodes in the generative Boltzmann machine. The open circles represent the results obtained by our method, where $K = 3$ and $\lambda = 2$. The closed squares represent the results obtained by the mean-field learning algorithm proposed by Welling and Hinton [9]. In this experiment, we stop updating by 500 times when the algorithms do not converge. Our method gives higher values of the log-likelihood than the method of Welling and Hinton. Their method significantly degrades at large J because it is based on a mean-field method, which often fails in highly correlated systems.

We show the dependency of the computational time of our learning algorithm on the number of hidden nodes, $|H|$. Figure 7 shows the average of computational time[‡] of our learning against the number of hidden nodes, where we set $|V| = 17$, $K = 3$, $\lambda = 2$, $h = 0.1$, and $J = 0.3$. The vertical axis represents the average of computational time which is required 500 update steps. The computational complexity of our method is $O(KM|L|)$ and $|L| = |V|(|V| - 1)/2 + |V||H| + |H| - 1 \propto |H|$ in the structure used in this section. Thus, the computational complexity of our method is

[‡] We use a 32 bit computer with 2.40 GHz CPU.

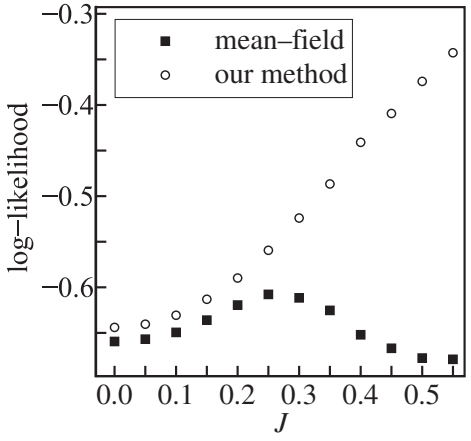

Fig. 6. Plot of the log-likelihood against J in the generative Boltzmann machine.

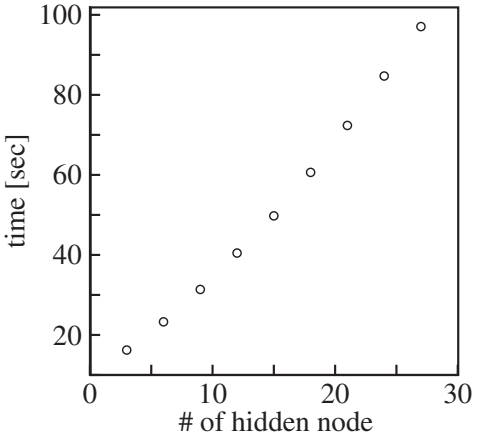

Fig. 7. The dependency of the computational time of our learning algorithm on $|H|$.

proportional to $|H|$. From Fig. 7, one can see that the computational time linearly grows with the increase in $|H|$ as we expect.

5. Conclusion

In this paper, we proposed a new learning algorithm for Boltzmann machines with hidden nodes using the MPA and MPLE, and we showed that our method is superior to the mean-field algorithm proposed Welling and Hinton [9]. Our framework allowed us to employ better likelihoods, e.g., composite likelihood [12], instead of pseudo-likelihood.

In the case of fully visible Boltzmann machines, the convergence of MPLE is theoretically guaranteed because the pseudo-likelihood function is a concave function. However, the convergence property of the proposed method remains unclear because of the existence of hidden nodes and probabilistic algorithms such as the MCMC method. The convergence property requires further investigation, which should be addressed in the future.

Acknowledgments

This work was partly supported by Grants-In-Aid (Nos. 21700247, 22300078, 24650115, and 24700220) for Scientific Research from the Ministry of Education, Culture, Sports, Science and Technology, Japan.

Appendix: Max-Product Algorithm in $P_{H|V}(\boldsymbol{x}_H \mid \boldsymbol{x}_V, \boldsymbol{\theta}, \boldsymbol{w})$

Given $\boldsymbol{x}_V$, $\boldsymbol{\theta}$, and $\boldsymbol{w}$, $P_{H|V}(\boldsymbol{x}_H \mid \boldsymbol{x}_V, \boldsymbol{\theta}, \boldsymbol{w})$ takes the form

$$P_{H|V}(\boldsymbol{x}_H \mid \boldsymbol{x}_V, \boldsymbol{\theta}, \boldsymbol{w}) \propto \exp\left(\sum_{i\in H} h_i x_i + \sum_{(i,j)\in L_H} w_{ij} x_i x_j\right) = \left(\prod_{i\in H} e^{h_i x_i}\right)\left(\prod_{(i,j)\in L_H} e^{w_{ij} x_i x_j}\right), \tag{A·1}$$

where $L_H := \{(i,j) \mid (i,j) \in L, i \in H, j \in H\}$ and

$$h_i := \theta_i + \sum_{j \in \partial_L(i) \cap V} w_{ij} x_j.$$

Therefore, we can regard $P_{H|V}(\boldsymbol{x}_H \mid \boldsymbol{x}_V, \boldsymbol{\theta}, \boldsymbol{w})$ as the Boltzmann machine on graph $G(H, L_H)$. By replacing H and L_H by Ω_0 and L_0, respectively, and by setting $\phi_i(x_i) = e^{h_i x_i}$ and $\psi_{ij}(x_i, x_j) = e^{w_{ij} x_i x_j}$, one identifies the distribution (A·1) using the Boltzmann machine (3.4). Hence, we can easily apply the MPA described in §3.2 to $P_{H|V}(\boldsymbol{x}_H \mid \boldsymbol{x}_V, \boldsymbol{\theta}, \boldsymbol{w})$.

REFERENCES

[1] Bishop, C. M., Pattern Recognition and Machine Learning, Springer (2006).

[2] Wainwright, M. J., and Jordan, M. I., "Graphical models, exponential families, and variational inference," *Foundations and Trends in Machine Learning*, **1**: 1–305 (2008).

[3] Ackley, D. H., Hinton, G. E., and Sejnowski, T. J., "A learning algorithm for Boltzmann machines," *Cognitive Science*, **9**: 147–169 (1985).

[4] MacKay, D. J., Information Theory, Inference, and Learning Algorithm, Cambridge University Press (2003).

[5] Hinton, G. E., "Training products of experts by minimizing contrastive divergence," *Neural Computation*, **14**: 1771–1800 (2002).

[6] Hinton, G. E., Osindero, S., and Teh, Y. W., "A fast learning algorithm for deep belief nets," *Neural Computation*, **18**: 1527–1554 (2006).

[7] Besag, J., "Statistical analysis of non-lattice data," *The Statistican*, **24**: 179–195 (1975).

[8] Hyvärinen, A., "Consistency of pseudo likelihood estimation of fully visible Boltzmann machines," *Neural Computation*, **18**: 2283–2292 (2006).

[9] Welling, M., and Hinton, G. E., "A new learning algorithm for mean field Boltzmann machines," *Proceeding of the International Conference on Artificial Neural Networks (ICANN)*, **2415**: 351–357 (2002).

[10] Carreira-Perpiñán, M. A., and Hinton, G. E., "On contrastive divergence learning," *Proceeding of 10th International Workshop on Artificial Intelligence and Statistics (AISTATS2005)*, 59–66 (2005).

[11] Marlin, B., Swersky, K., Chen, B., and de Freitas, N., "Inductive principles for restricted Boltzmann machine learning," *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics (AISTATS)*, **9**: 509–516 (2010).

[12] Lindsay, B. G., "Composite likelihood methods," *Contemporary Mathematics*, **80**: 221–239 (1988).