IEICE TRANS. INF. & SYST., VOL.E101–D, NO.2 FEBRUARY 2018

PAPER

Deep Relational Model: A Joint Probabilistic Model with a Hierarchical Structure for Bidirectional Estimation of Image and Labels

Toru NAKASHIKA[†a)], ***Member***

SUMMARY Two different types of representations, such as an image and its manually-assigned corresponding labels, generally have complex and strong relationships to each other. In this paper, we represent such *deep* relationships between two different types of visible variables using an energy-based probabilistic model, called a deep relational model (DRM) to improve the prediction accuracies. A DRM stacks several layers from one visible layer on to another visible layer, sandwiching several hidden layers between them. As with restricted Boltzmann machines (RBMs) and deep Boltzmann machines (DBMs), all connections (weights) between two adjacent layers are undirected. During maximum likelihood (ML) -based training, the network attempts to capture the latent complex relationships between two visible variables with its deep architecture. Unlike deep neural networks (DNNs), 1) the DRM is a totally generative model and 2) allows us to generate one visible variables given the other, and 2) the parameters can be optimized in a probabilistic manner. The DRM can be also fine-tuned using DNNs, like deep belief nets (DBNs) or DBMs pre-training. This paper presents experiments conduced to evaluate the performance of a DRM in image recognition and generation tasks using the MNIST data set. In the image recognition experiments, we observed that the DRM outperformed DNNs even without fine-tuning. In the image generation experiments, we obtained much more realistic images generated from the DRM more than those from the other generative models.

key words: *image classification, image generation, deep learning, generative model, Boltzmann distribution*

1. Introduction

Since Hinton *et al.* introduced an effective pre-training algorithm for deep neural networks* (DNNs) using deep belief networks (DBNs) in 2006 [1], the use of deep learning has rapidly spread in the field of machine learning, artificial intelligence, signal processing, etc. A DBN is a graphical model that stacks restricted Boltzmann machines (RBMs) [2], [3] layer-by-layer, each of which represents the probability distribution of visible variables with hidden variables. The effectiveness of using DBNs (or RBMs) has been proved especially in discriminative or deterministic tasks, such as handwritten character recognition [1], 3-D object recognition [4], machine transliteration [5], speech recognition [6], and voice conversion [7]. The discriminative tasks are generally achieved by setting the initial values of weights of a DNN as the trained weights of a DBN, and running back-propagation to fine-tune the DNN weights. This can be done due to the ability of deep learning that captures high-level abstractions at higher layers.

When it comes to the use of deep learning for generation tasks, we can find various models, such as a deep Boltzmann machines (DBM) [8], [9], a denoising auto-encoder (DAE) [10], a shape Boltzmann machine (ShapeBM) [11], and a sum-product network (SPN) [12]. These models were mainly introduced to capture high-order abstractions for good representation of the observations, rather than for discriminative goal. Once obtaining high-level abstractions, we can, for instance, remove some noise on the observations, or restore missing parts in the observations.

Most of the existing deep-learning approaches focus on extracting high-order abstractions from one variable. In this paper, we try to capture such high-order relationships between two different types of variables based on deep learning. For that, we introduced a probabilistic model called a deep relational model (DRM) [13]. A DRM is similar to an RBM and a DBM, each of which is a probabilistic model based on an energy function. The model sandwiches several hidden layers** between two visible layers and defines a joint probability for the two visible variables. Every two adjacent layers are connected with undirected weights, which are estimated so as to maximize the likelihood of the two visible variables. Interestingly, since the DRM is a totally generative model, it allows us not only to apply it to recognition tasks, but to also generate samples of one variable from the other variable. For example, considering that we have two kinds of variables for a hand-written digit image and a one-hot vector of the labels, we can estimate the label by inferring mean-field posteriors given an image (classification task). On the other hand, by inferring posteriors given a label, we could obtain a generated image corresponding to the label (generation task). In this paper, we report additional experimental results to further investigate the performance of the DRM.

This paper is organized as follows. In Sect. 2, we state

Manuscript received May 3, 2017.
Manuscript revised September 18, 2017.
Manuscript publicized October 25, 2017.

[†]The author is with the Graduate School of Informatics and Engineering, University of Electro-Communications, Chofu-shi, 182–8585 Japan.
a) E-mail: nakashika@uec.ac.jp

DOI: 10.1587/transinf.2017EDP7149

*The term "neural networks" usually refers to a feedforward (directed) type of neural networks, and we also follow this here.

**When we give one hidden layer for our model, it is equivalent to an RBM with a concatenated vector of two visible variables. This will be discussed later.

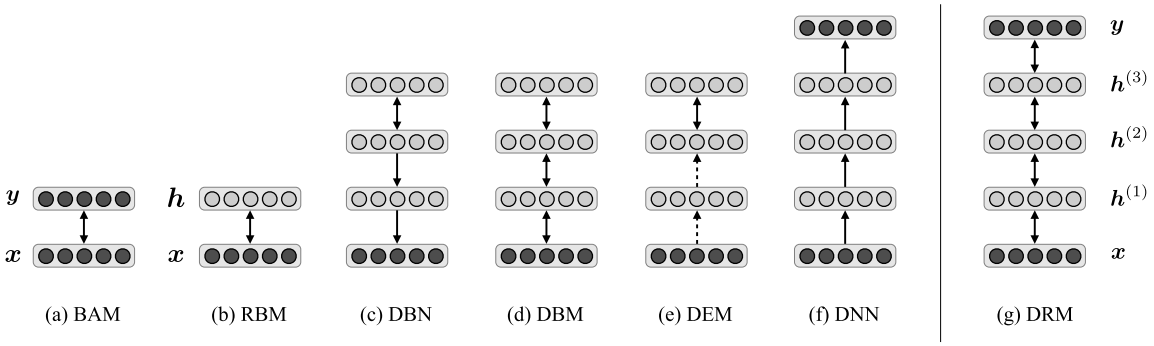

Fig. 1 Graphical representation of (a) bidirectional associative memories, (b) a restricted Boltzmann machine, (c) a deep belief network, (d) a deep Boltzmann machine, (e) a deep energy model, (f) a deep neural network, and (g) a deep relational model. Two-way arrows and one-way arrows indicate undirected weights and directed weights, respectively. Dotted arrows represent deterministic relationships.

the differences between the DRM and related models. In Sect. 3, we review the formulation of energy-based models. We show the definition of a DRM and its parameter estimation algorithm in Sect. 4. In Sect. 5, we show our experimental results and conclude our findings in Sect. 6.

2. Related Work

The purpose of this paper is to improve the prediction accuracies (such as classification error rate) by means of the cyclic bidirectional propagation between two different visible variables, $\boldsymbol{x}$ and $\boldsymbol{y}$. When we predict $\boldsymbol{y}$ from $\boldsymbol{x}$, measuring the closeness of the input $\boldsymbol{x}$ and the reconstruction of $\boldsymbol{x}$ from the predicted $\boldsymbol{y}$ would also help the further prediction of $\boldsymbol{y}$. Collaterally, we can also use the already-trained model for the reverse prediction; i.e., the prediction of $\boldsymbol{x}$ from $\boldsymbol{y}$, which reduce the additional costs to train another model for the reverse prediction. In this section, we compare our proposed model, a deep relational model (DRM), with other related models: bidirectional associative memories (BAMs) [14], a restricted Boltzmann machine (RBM), a deep belief network (DBN) [1], a deep Boltzmann machine (DBM) [8], [9], a deep energy model (DEM) [15], and a deep neural network (DNN). These models are graphically represented in Fig. 1. BAMs and an RBM consist of two layers with having bidirectional connections between them. The difference of these is that BAMs represent the relationships between two different visible variables $\boldsymbol{x}$ and $\boldsymbol{y}$, while an RBM represents one visible variable $\boldsymbol{x}$ and hidden variable $\boldsymbol{h}$. As shown in Fig. 1, each model other than BAMs and an RBM has a deep architecture by stacking a visible layer $\boldsymbol{x}$ and multiple hidden layers $\boldsymbol{h}_1, \boldsymbol{h}_2, \cdots$ layer-by-layer with having unidirectional or bidirectional connections between adjacent two layers. The deep architecture has the capability of representing more complex data, compared with an RBM that stacks a single hidden layer. A DNN and the proposed model further stack another visible variable $\boldsymbol{y}$ on the top. Therefore, these two models try to capture latent relationships between $\boldsymbol{x}$ and $\boldsymbol{y}$, while the other models just discover latent features or representation from $\boldsymbol{x}$. In classification tasks, the RBMs were used as classifiers by splitting the visible units into classification units and observation units [16], [17]. These approaches can be regarded as one hidden layer version of the proposed model; in other words, the DRM method extends the classification by means of the RBMs.

An important factor in distinguishing each model is the direction of the connections between two adjacent layers. For example, a DBN has undirected connections at the top two layers, which form an RBM, and directed connections to the lower layers. A general DNN is a feedforward model; every two adjacent layers have deterministic weights in the direction from the source to the target variables. Meanwhile, the proposed DRM has totally bidirectional connections through all layers, just like a DBM does. This leads to the propagation of information from the bottom up and from the top down in the network, while a DNN only infers from bottom to top. Assuming $\boldsymbol{x}$ and $\boldsymbol{y}$ indicate a vectorized image and a one-hot vector of the labels, a DRM allows us not only to estimate the label vector given an image, but also to generate an image from given a label vector.

Another aspect is the way parameters are estimated. Energy-based models, which include BAMs, RBMs, DBMs, DEMs, and DRMs, are stochastic models in which the parameters are estimated so as to maximize the likelihood of observations[†]. On the other hand, the parameters of a DNN are optimized in a deterministic manner to minimize the mean square error (MSE) or the cross entropy (CE) using a back-propagation algorithm. Since a stochastic model, such as a DRM, optimizes the parameters in a probabilistic framework, we can further extend the parameter estimation method to using maximum a posteriori (MAP), Bayesian inference, and so on.

Typically, deep-learning methods, such as a DBN, a DBM and a DEM, are used for the pre-training of a DNN. As reported in [1], a pre-trained DNN dramatically outperformed a randomly-initialized DNN. Generally speaking, in a deep network, error signals get weaker as they are back-propagated to the lower layer, which causes difficulties in estimating the parameters of the lower layer. Therefore,

[†]In practice, an approximation method is used.

the pre-training approaches are considered to be effective in compensating for the *thin* gradients of the parameters. However, these approaches learn high-order representation in an unsupervised manner without knowing the existence of the target features. Therefore, it could be said that the learned weights are not necessarily appropriate for the initial values of a DNN that takes the target features into account. Our model, in contrast, connects with a visible layer for the target features and optimizes the parameters jointly, which may lead to better results compared with the above methods, even in a recognition task. Furthermore, our model is not adversely affected by the problems associated with DBMs. During the training of a DBM, it is difficult to estimate the weight parameters at the higher layers due to the fading gradients far from the visible layer [9]. On the contrary, our model sandwiches hidden layers with two visible layers at the opposite sides, and hence it propagates gradients more clearly top-to-bottom and bottom-to-top.

As for a DEM, Ngiam *et al.* also proposed a discriminative extension that considers target features in the model [15]. The model is, however, still discriminative; it does not have an ability to generate the source features from the target features. Furthermore, what the weights at the lower layers are trained without knowing about the target features also applies to this model.

3. Energy-Based Models

Our model, a deep relational model (DRM), will be defined as an energy-based model. In this section, we briefly review energy-based models and remind of some kinds.

Energy-based models gives an energy to each configuration of the variables, such as an energy of a single unit of the variables (unary potential), and an energy between two units of the variables (pair-wise potential). These kinds of probabilistic models define a probability density function (PDF) using an arbitrary energy function $E(\boldsymbol{x};\theta)$, as follows:

$$p(\boldsymbol{x};\theta) = \frac{1}{Z(\theta)} e^{-E(\boldsymbol{x};\theta)}, \tag{1}$$

where Z is a normalization term so that the summation of the probability over $\boldsymbol{x}$ equals to 1 (i.e., $Z = \sum_{\boldsymbol{x}} e^{-E(\boldsymbol{x};\theta)}$), and θ is model parameters to be estimated. Note that Z is a function that depends on not $\boldsymbol{x}$ but θ.

The parameters of an energy-based model can be estimated by performing stochastic gradient descent (SGD) on the log-likelihood of the training data (N samples). Specifically, the objective function is as follows:

$$\mathcal{L}(\theta;\mathcal{D}) = \frac{1}{N} \sum_{\boldsymbol{x}\in\mathcal{D}} \log p(\boldsymbol{x};\theta). \tag{2}$$

Using the stochastic gradient, which is calculated as

$$\frac{\partial \mathcal{L}(\theta;\mathcal{D})}{\partial \theta} = -\frac{1}{N} \sum_{\boldsymbol{x}\in\mathcal{D}} \frac{\partial E(\boldsymbol{x};\theta)}{\partial \theta} + \sum_{\tilde{\boldsymbol{x}}} p(\tilde{\boldsymbol{x}}) \frac{\partial E(\tilde{\boldsymbol{x}};\theta)}{\partial \theta}, \tag{3}$$

each paramter is iteratively updated as follows:

$$\theta^{(\text{new})} = \theta^{(\text{old})} - \eta \frac{\partial \mathcal{L}(\theta^{(\text{old})};\mathcal{D})}{\partial \theta^{(\text{old})}}, \tag{4}$$

where η is a learning rate and empirically determined. However, it is usually difficult to compute the second term in Eq. (3) due to enormous amount of calculation of all possible configurations. Therefore, a sampling method such as Monte-Carlo, Gibbs sampling [18], or contrastive divergence [1] is usually used to approximate the second term. For more efficient learning, we can also employ the adaptive learning rate [19] or parallel tempering learning methods [20], [21].

3.1 Restricted Boltzmann Machine

A restricted Boltzmann machine (RBM) [2], [3] is one of the energy-based models, which models a joint probability distribution of visible binary-variables $\boldsymbol{x} \in \{0,1\}^I$ and invisible (hidden) binary-variables $\boldsymbol{h} \in \{0,1\}^J$ as shown in Fig. 1 (b). In this model, it is assumed that there are undirected connections between visible-hidden units but no connections between visible-visible units nor hidden-hidden units. The probability distribution is defined as:

$$p(\boldsymbol{x};\theta) = \sum_{\boldsymbol{h}} p(\boldsymbol{x},\boldsymbol{h};\theta) \tag{5}$$

$$p(\boldsymbol{x},\boldsymbol{h};\theta) = \frac{1}{Z(\theta)} e^{-E_{\text{RBM}}(\boldsymbol{x},\boldsymbol{h};\theta)}, \tag{6}$$

with the following energy function:

$$E_{\text{RBM}}(\boldsymbol{x},\boldsymbol{h};\theta) = -\boldsymbol{b}^\top \boldsymbol{x} - \boldsymbol{c}^\top \boldsymbol{h} - \boldsymbol{x}^\top \mathbf{W} \boldsymbol{h}, \tag{7}$$

where $\mathbf{W} \in \mathbb{R}^{I\times J}$, $\boldsymbol{b} \in \mathbb{R}^I$, and $\boldsymbol{c} \in \mathbb{R}^J$ are model parameters for the weights of connection between visible units and hidden units, a bias vector of the visible units, and a bias vector of the hidden units, respectively.

Because neither visible nor hidden units are connected to each other, the conditional probabilities $p(\boldsymbol{x}|\boldsymbol{h})$ and $p(\boldsymbol{h}|\boldsymbol{x})$ form simple equations as follows:

$$p(x_i|\boldsymbol{h}) = \mathcal{B}(x_i;\sigma(b_i + \mathbf{W}_{i:}\boldsymbol{h})) \tag{8}$$

$$p(h_j|\boldsymbol{x}) = \mathcal{B}(\sigma(c_j + \mathbf{W}_{:j}^\top \boldsymbol{x})), \tag{9}$$

where $\mathbf{W}_{i:}$ and $\mathbf{W}_{:j}$ denote the ith row and the jth column vectors of the matrix $\mathbf{W}$, respectively. $\mathcal{B}(\cdot;\pi)$ and $\sigma(\cdot)$ indicate the Bernoulli distribution with the success probability π and an element-wise sigmoid function that is $\sigma(\boldsymbol{x}) = \frac{1}{1+e^{-x}}$, respectively.

3.2 Deep Boltzmann Machine

Another example of the energy-based model is a deep Boltzmann machine (DBM) [8], [9]. A DBM stacks multiple hidden layers with having undirected connections through all layers as shown in Fig. 1 (d). The deep architecture may help to capture more complicated, higher-order internal representations. A generall form of the DBM that

consists of visible variables $\boldsymbol{x} \in \{0,1\}^I$ and L hidden variables $\boldsymbol{h}^{(l)} \in \{0,1\}^{J_l}$ $(l = 1, \cdots, L)$ defines the probability distribution as follows:

$$p(\boldsymbol{x};\theta) = \sum_{\forall \boldsymbol{h}^{(l)}} p(\boldsymbol{x}, \forall \boldsymbol{h}^{(l)};\theta) \tag{10}$$

$$p(\boldsymbol{x}, \forall \boldsymbol{h}^{(l)};\theta) = \frac{1}{Z(\theta)} e^{-E_{\mathrm{DBM}}(\boldsymbol{x}, \forall \boldsymbol{h}^{(l)};\theta)}. \tag{11}$$

The energy function is defined as:

$$\begin{aligned} E_{\mathrm{DBM}}(\boldsymbol{x}, \forall \boldsymbol{h}^{(l)};\theta) &= -\boldsymbol{b}^\top \boldsymbol{x} - \sum_{l=1}^{L} \boldsymbol{c}^{(l)\top} \boldsymbol{h}^{(l)} \\ &\quad - \boldsymbol{x}^\top \mathbf{W}^{(1)} \boldsymbol{h}^{(1)} - \sum_{l=2}^{L} \boldsymbol{h}^{(l-1)\top} \mathbf{W}^{(l)} \boldsymbol{h}^{(l)}, \end{aligned} \tag{12}$$

where $\mathbf{W}^{(l)} \in \mathbb{R}^{J_{l-1} \times J_l}$ and $\boldsymbol{c}^{(l)} \in \mathbb{R}^{J_l}$ are additional parameters to the RBM parameters.

The conditional probabilities are given by:

$$p(x_i|\boldsymbol{h}^{(1)}) = \mathcal{B}(x_i; \sigma(b_i + \mathbf{W}^{(1)}_{i:} \boldsymbol{h}^{(1)})) \tag{13}$$

$$\begin{aligned} &p(h^{(l)}_j|\boldsymbol{h}^{(l-1)}, \boldsymbol{h}^{(l+1)}) \\ &\quad = \mathcal{B}(h^{(l)}_j; \sigma(c^{(l)}_j + \mathbf{W}^{(l)\top}_{:j} \boldsymbol{h}^{(l-1)} + \mathbf{W}^{(l+1)}_{j:} \boldsymbol{h}^{(l+1)})) \end{aligned} \tag{14}$$

$$p(h^{(L)}_j|\boldsymbol{h}^{(L-1)}) = \mathcal{B}(h^{(L)}_j; \sigma(c^{(L)}_j + \mathbf{W}^{(L)\top}_{:j} \boldsymbol{h}^{(L)})). \tag{15}$$

Note that the middle hidden layers take values from two layers as shown in Eq. (14).

4. Deep Relational Model

Considering a dataset of images and its the labels, the labels should have been intentionally-, carefully-, and manually-assigned. As a result, there must be a strong correlation between an image and the assigned label. To capture latent, complicated, high-order relationships between two observable variables, such as an image and a one-hot vector of the label, we introduce a deep stochastic network called a deep relational model (DRM).

4.1 Definition and Generative Procedure

As shown in Fig. 1 (g), a DRM is a deep network that sandwiches multiple hidden layers with two visible layers. As an energy-based model, a DRM defines a joint probability distribution of one (first) visible variables $\boldsymbol{x} \in \{0,1\}^I$ and the other (second) visible variables $\boldsymbol{y} \in \{0,1\}^K$ along with hidden variables $\boldsymbol{h}^{(l)} \in \{0,1\}^{J_l}$ $(l = 1, \cdots, L)$, where L is the number of hidden layers. Similarly to an RBM and a DRM, each unit is only connected to the units at the adjacent layers, and is not connected to the units at the same layer. We define the joint probability distribution using a DRM as follows:

$$p(\boldsymbol{x}, \boldsymbol{y};\theta) = \sum_{\forall \boldsymbol{h}^{(l)}} p(\boldsymbol{x}, \boldsymbol{y}, \forall \boldsymbol{h}^{(l)};\theta) \tag{16}$$

$$p(\boldsymbol{x}, \boldsymbol{y}, \forall \boldsymbol{h}^{(l)};\theta) = \frac{1}{Z(\theta)} e^{-E_{\mathrm{DRM}}(\boldsymbol{x}, \boldsymbol{y}, \forall \boldsymbol{h}^{(l)};\theta)}, \tag{17}$$

where the energy function E_{DRM} is defined as:

$$\begin{aligned} &E_{\mathrm{DRM}}(\boldsymbol{x}, \boldsymbol{y}, \forall \boldsymbol{h}^{(l)};\theta) \\ &\quad = -\boldsymbol{b}^\top \boldsymbol{x} - \sum_{l=1}^{L} \boldsymbol{c}^{(l)\top} \boldsymbol{h}^{(l)} - \boldsymbol{d}^\top \boldsymbol{y} - \boldsymbol{x}^\top \mathbf{W}^{(1)} \boldsymbol{h}^{(1)} \\ &\qquad - \sum_{l=2}^{L} \boldsymbol{h}^{(l-1)\top} \mathbf{W}^{(l)} \boldsymbol{h}^{(l)} - \boldsymbol{h}^{(L)\top} \mathbf{W}^{(L+1)} \boldsymbol{y}. \end{aligned} \tag{18}$$

In addition to the previously-defined parameters $\boldsymbol{b}$, $\boldsymbol{c}^{(l)}$, and $\mathbf{W}^{(l)}$, the bias parameters for the second visible variables $\boldsymbol{d} \in \mathbb{R}^K$ are used. $\mathbf{W}^{(L+1)} \in \mathbb{R}^{J_l \times K}$ is the connection weights between the highest hidden layer and the second visible layer.

Each conditional distributions given the units at the adjacent layers can be computed as:

$$p(x_i|\boldsymbol{h}^{(1)}) = \mathcal{B}(x_i; \sigma(b_i + \mathbf{W}^{(1)}_{i:} \boldsymbol{h}^{(1)})) \tag{19}$$

$$\begin{aligned} &p(h^{(l)}_j|\boldsymbol{h}^{(l-1)}, \boldsymbol{h}^{(l+1)}) \\ &\quad = \mathcal{B}(h^{(l)}_j; \sigma(c^{(l)}_j + \mathbf{W}^{(l)\top}_{:j} \boldsymbol{h}^{(l-1)} + \mathbf{W}^{(l+1)}_{j:} \boldsymbol{h}^{(l+1)})) \end{aligned} \tag{20}$$

$$p(y_k|\boldsymbol{h}^{(L)}) = \mathcal{B}(y_k; \sigma(d_k + \mathbf{W}^{(L)\top}_{:k} \boldsymbol{h}^{(L)})). \tag{21}$$

Note that the conditional probabilities of $\boldsymbol{h}^{(1)}$ and $\boldsymbol{h}^{(L)}$ can be calculated from Eq. (20) by regarding as $\boldsymbol{h}^{(0)} = \boldsymbol{x}$ and $\boldsymbol{h}^{(L+1)} = \boldsymbol{y}$, respectively. Although the joint configuration of $\boldsymbol{x}$ and $\boldsymbol{y}$ is defined in a DRM, the first variable $\boldsymbol{x}$ is not directly connected to the second variable $\boldsymbol{y}$, and $\boldsymbol{y}$ is not required to infer $\boldsymbol{x}$, as Eq. (19) indicates. Through hidden layers, $\boldsymbol{x}$ and $\boldsymbol{y}$ propagate their information to each other layer-by-layer. Therefore, the network models deep latent correlations between $\boldsymbol{x}$ and $\boldsymbol{y}$. That means the trained network has the ability to estimate one variable given the other variable. To estimate variable $\hat{\boldsymbol{y}}$ given $\boldsymbol{x}$, for example, we use an iterative mean-field update approach, as shown in Fig. 2. In this procedure, we first compute the expectations (mean-field approximation) for each hidden layer's unit from bottom to top, as in Eq. (20), regarding all the values of the units at the upper layer as zero. Then, we calculate the expectations of hidden units using the previously-calculated values

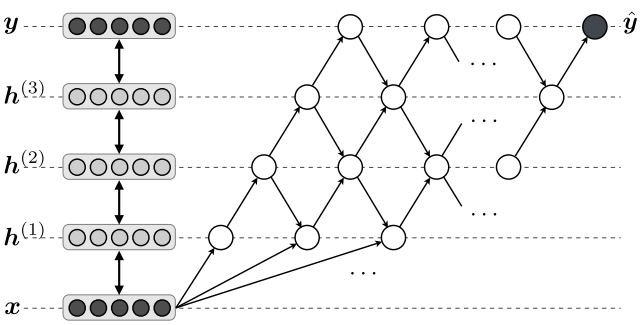

Fig. 2 Generating $\hat{\boldsymbol{y}}$ from $\boldsymbol{x}$ by repeating mean-field updates.

for $\boldsymbol{h}^{(l-1)}$ and $\boldsymbol{h}^{(l)}$ in Eq. (20). We iterate this procedure T times with clamping the values of $\boldsymbol{x}$ (in our experiments, we used $T = 100$). Finally, we obtain the expected values of $\boldsymbol{y}$ by calculating $\mathbb{E}[\boldsymbol{y}|\boldsymbol{x}] \approx \mathbb{E}[\boldsymbol{y}|\hat{\boldsymbol{h}}^{(L)}] = \sigma(\boldsymbol{d} + \mathbf{W}^{(L)\top}\hat{\boldsymbol{h}}^{(L)})$, where $\hat{\boldsymbol{h}}^{(L)}$ is the lastly-updated $\hat{\boldsymbol{h}}^{(L)}$ after the iteration.

We can also extend[†] the DRM so that it feeds real-valued data for $\boldsymbol{x}$ and/or $\boldsymbol{y}$ using the Gaussian scheme like Gaussian-Bernoulli RBM [22] or Gaussian-Bernoulli DBM [23]. In this scheme, when we want to feed real-valued $\boldsymbol{x} \in \mathbb{R}^I$, we replace the $\boldsymbol{x}$-related terms in Eq. (18) $-\boldsymbol{b}^\top\boldsymbol{x} - \boldsymbol{x}^\top\mathbf{W}^{(1)}\boldsymbol{h}^{(1)}$ with $\boldsymbol{x}^\top\boldsymbol{\Sigma}_x\boldsymbol{x}/2 - \boldsymbol{b}^\top\boldsymbol{\Sigma}_x\boldsymbol{x} - \boldsymbol{x}^\top\boldsymbol{\Sigma}_x\mathbf{W}^{(1)}\boldsymbol{h}^{(1)}$, where $\boldsymbol{\Sigma}_x \triangleq \mathrm{diag}(\boldsymbol{s}_x^2)$ indicates the diagonal matrix whose diagonal elements are variances of $\boldsymbol{x}$, $\boldsymbol{s}_x^2 \in \mathbb{R}^I$. This changes the conditional probability $p(x_i = 1|\boldsymbol{h}^{(1)})$ in Eq. (19) as follows:

$$p(x_i|\boldsymbol{h}^{(1)}) = \mathcal{N}(x_i; b_i + \mathbf{W}^{(1)}_{i:}\boldsymbol{h}^{(1)}, {s_x}_i^2) \tag{22}$$

where $\mathcal{N}(\cdot;\mu,s^2)$ indicates the Gaussian distribution with the mean μ and the variance s^2. For the real-valued $\boldsymbol{y} \in \mathbb{R}^K$, we can similarly modify the definition in Eq. (18) by replacing $-\boldsymbol{d}^\top\boldsymbol{y} - \boldsymbol{h}^{(L)\top}\mathbf{W}^{(L+1)}\boldsymbol{y}$ with $\boldsymbol{y}^\top\boldsymbol{\Sigma}_y\boldsymbol{y}/2 - \boldsymbol{d}^\top\boldsymbol{\Sigma}_y\boldsymbol{y} - \boldsymbol{h}^{(L)\top}\mathbf{W}^{(L+1)}\boldsymbol{\Sigma}_y\boldsymbol{y}$, where $\boldsymbol{\Sigma}_y \triangleq \mathrm{diag}(\boldsymbol{\sigma}_y^2)$, $\boldsymbol{\sigma}_y^2 \in \mathbb{R}^K$, which yields the following conditional probability:

$$p(y_k|\boldsymbol{h}^{(L)}) = \mathcal{N}(y_k; d_k + \mathbf{W}^{(L)\top}_{:k}\boldsymbol{h}^{(L)}, {\sigma_y}_k^2). \tag{23}$$

4.2 Parameter Optimization

For parameter estimation, the joint log-likelihood of $\boldsymbol{x}$ and $\boldsymbol{y}$, $\mathcal{L}(\theta;\mathcal{D}) = \frac{1}{N}\sum_{(\boldsymbol{x},\boldsymbol{y})\in\mathcal{D}} \log p(\boldsymbol{x},\boldsymbol{y};\theta)$, is used. Partially differentiating the likelihood with respect to each parameter, we obtain:

$$\frac{\partial\mathcal{L}(\theta;\mathcal{D})}{\partial b_i} = \langle x_i\rangle_{\mathrm{d}} - \langle x_i\rangle_{\mathrm{m}} \tag{24}$$

$$\frac{\partial\mathcal{L}(\theta;\mathcal{D})}{\partial c_j^{(l)}} = \langle h_j^{(l)}\rangle_{\mathrm{d}} - \langle h_j^{(l)}\rangle_{\mathrm{m}} \tag{25}$$

$$\frac{\partial\mathcal{L}(\theta;\mathcal{D})}{\partial d_k} = \langle y_k\rangle_{\mathrm{d}} - \langle y_k\rangle_{\mathrm{m}} \tag{26}$$

$$\frac{\partial\mathcal{L}(\theta;\mathcal{D})}{\partial W_{ij}^{(l)}} = \begin{cases} \langle x_i h_j^{(1)}\rangle_{\mathrm{d}} - \langle x_i h_j^{(1)}\rangle_{\mathrm{m}} & (l=1) \\ \langle h_i^{(l-1)} h_j^{(l)}\rangle_{\mathrm{d}} - \langle h_i^{(l-1)} h_j^{(l)}\rangle_{\mathrm{m}} & (l = 2,\cdots,L) \\ \langle h_i^{(L)} y_j\rangle_{\mathrm{d}} - \langle h_i^{(L)} y_j\rangle_{\mathrm{m}} & (l = L+1) \end{cases} \tag{27}$$

where $\langle\cdot\rangle_{\mathrm{d}}$ and $\langle\cdot\rangle_{\mathrm{m}}$ indicate the expectations of the empirical data and the inner model, respectively. As mentioned before, the second terms are computationally difficult. Therefore, we approximate the second terms with the expectations of the reconstructed data $(\bar{\boldsymbol{x}},\bar{\boldsymbol{y}})$ that are sampled from the iteratively-updated inner model (Fig. 3). The iterative procedure is similar to the generation scheme shown in Fig. 2,

[†]Nevertheless, in this paper, we focus on the evaluation of Bernoulli-Bernoulli DRM.

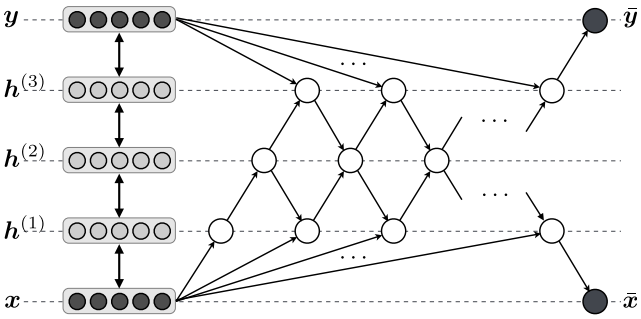

Fig. 3 Iterative inference using mean-field updates. White circles and black circles indicate mean-field inference and randomly-generated samples with their probabilities, respectively.

but we use the empirical values of $\boldsymbol{y}$ during iteration. After updating each expected value of hidden units T times, we sample $\bar{\boldsymbol{x}}$ and $\bar{\boldsymbol{y}}$ using Eqs. (19) and (21).

In our preliminary experiments, we observed that all the parameters of a DRM can be simultaneously estimated using the above iteration procedure starting from randomly-initialized values. However, to boost up parameter optimization and avoid the local maxima problem, we can also employ a pre-training scheme illustrated in Fig. 4 in practice. In this scheme, before training a DRM, we perform greedy layer-wise training, which is equivalent to a DBN [1]. That is, we first train the RBM with the visible units of one representation (in Fig. 4, $\boldsymbol{x}$), then train the following RBM by setting the visible units with the expected values of the hidden units inferred from the previous RBM, and repeat this procedure until obtaining the last hidden layer. In the training stage of the DRM, the parameters obtained in the pre-training are set to the initial values of the training.

When we want to do image recognition tasks, we can estimate the label $\boldsymbol{y}$ given an image $\boldsymbol{x}$, as discussed in the previous subsection (see Fig. 2). However, we can also employ a fine-tuning scheme. As shown in the right of Fig. 4, after the training of the DRM, we fine-tune each parameter using back-propagation, treating it as a discriminative DNN.

5. Experiments

5.1 Setup

To evaluate our method and examine its potential, we conducted recognition and generation experiments using the MNIST dataset. The dataset contains 60,000 training and 10,000 test images of handwritten digits (0–9) with a size of 28×28 pixels, along with the manually-assigned label data. To speed-up learning, we divided the training data into mini-batches, each of which contained 50 data, and trained the model with the fixed learning rate of 0.01 in 100 epochs. For the network architecture, we followed the configuration in [8], which led to preferable results; i.e., we used the four-layer network architecture consisting of the first visible layer of 784 units, the first hidden layer of 500 units, the second hidden layer of 1,000 units, and the second visible layer of

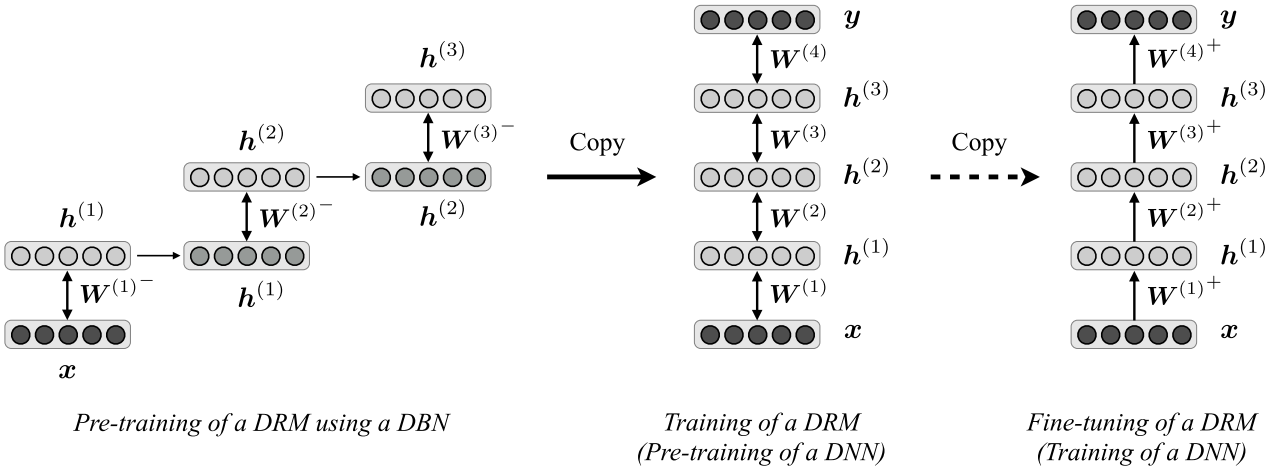

Fig. 4 Flow of training a DRM (in this example, three-hidden-layer DRM). Firstly, the weight parameters are pre-trained by performing the training of a DBN. Secondly, the parameters are optimized in a whole network of a DRM. When we apply the model to discriminative tasks, the DRM parameters are used as the initial values of a DNN, and then fine-tuned using back-propagation.

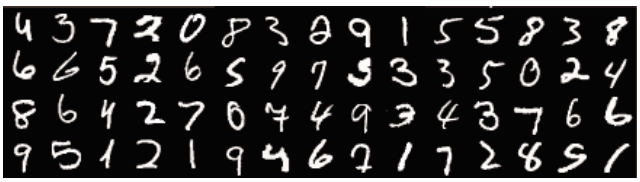

Fig. 5 Examples from the MNIST dataset.

10 units.

5.2 Convergence Curve with/without Pre-Training

Before going into details of the evaluation, we investigated the effectiveness of the proposed pre-training method of DRM discussed in Sect. 4.2. Figure 6 compares the convergence curves when using the pre-training scheme (lines in red) and when training the DRM without pre-training (lines in blue). As shown in Fig. 6, the training of DRM was pretty stable even without pre-training. However, the pre-training scheme avoided local maxima and considerably improved the both MSE curves in terms of convergence speed and accuracy. As for the evaluation to the test set, we obtained the error rates of 1.03% and 1.27% from the DRM with and without pre-training, respectively. This shows the importance of introducing the pre-training.

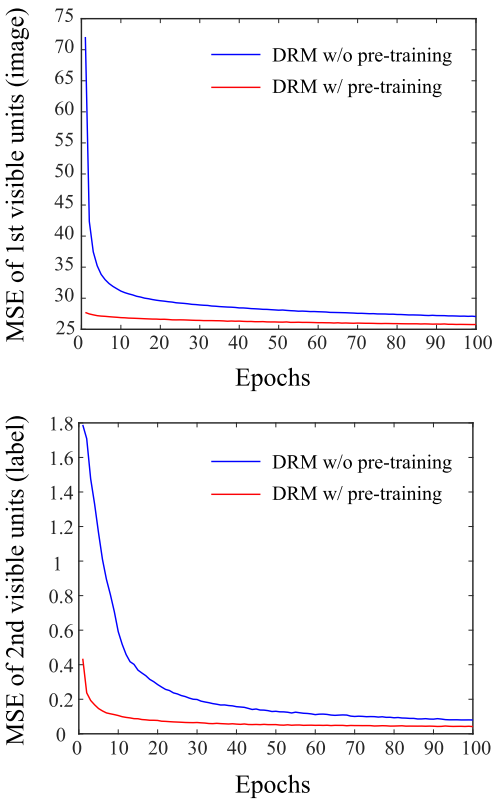

Fig. 6 Convergences curve of DRM with and without pre-training regarding the first visible units $\boldsymbol{x}$ (above) and the second visible units $\boldsymbol{y}$ (below). The vertical and horizontal axes indicate the MSE and the number of epochs during the training, respectively.

5.3 Results and Discussion

5.3.1 Classification Task

First, we compared our model, DRM, with the conventional DNN in image classification by changing the number of training data as $1k$, $10k$, $30k$, and $60k$. We used the same network architecture of [784-500-1000-10] for the DNN. The reason to compare with the randomly-initialized DNN is because the most current applications based on deep learning use the simple DNN without pre-training [24]. We also compared the DRM with and without fine-tuning (these will be identified as "DRM" and "fine-DRM," respectively). Each configuration was repeated five times and evaluated with the average error rate and the 95% confidence intervals because the methods use the SGD optimization that includes ran-

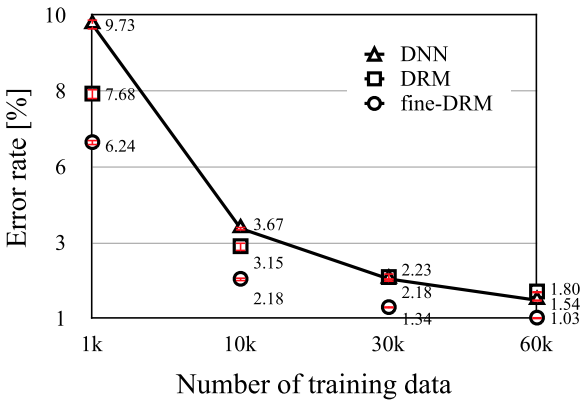

Fig. 7 Error rate [%] from a DNN, a DRM, and its fine-tuned. The error bars in red indicate the 95% confidence intervals through five trials.

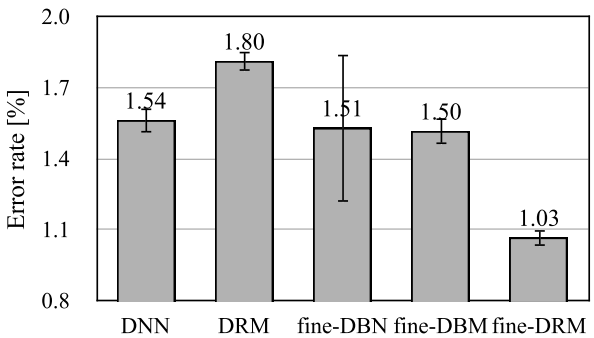

Fig. 8 Error rate [%] for the MNIST dataset obtained by each method. The error bars indicate the 95% confidence intervals through five trials.

dom permutation. The results are shown in Fig. 7, which indicates that the fine-DRM significantly outperformed the DNN regardless of the number of training data. It is worth noting that we obtained better performance from the DRM even without fine-tuning than the DNN when the training size was small, and similar performance from the DNN and the DRM when the training size was large, although the DRM without fine-tuning is generative model while the DNN is discriminative model. These results are similar to those in [16], which describes that the smaller training sets tend to favor generative learning. As shown in Fig. 7, the differences in performance of each method becomes more significant as the smaller amount of training data is used.

Secondly, we compared our method with the three conventional methods: a DNN, a DBN, and a DBM with the same condition of the DRM when all the training data was given. Each method had the same network architecture ([784-500-1000-10]) and evaluated as average error rate through five trials. After training the DBN and the DBM, we fine-tuned their parameters using back-propagation (noted as "fine-DBN" and "fine-DBM", respectively, in Fig. 8), while a DNN trained the parameters starting from randomly-initialized values. Figure 8 shows the comparison results. Our model "DRM" used the mean-field-update scheme to estimate $\boldsymbol{y}$ (Fig. 2), and "fine-DRM" used the fine-tuning scheme and produced the label vectors in a feedforward

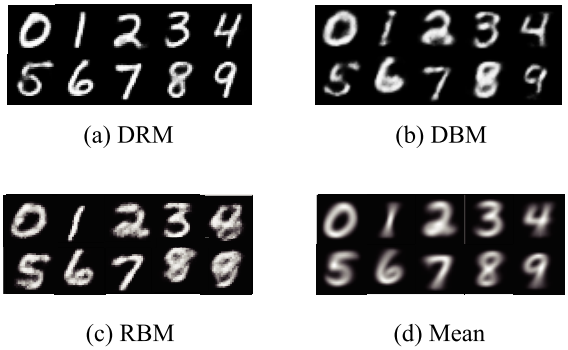

Fig. 9 Generated images given each one-hot label using DRM (a), DBM (b), and RBM (c), and mean values over the training data calculated for each label (d).

DNN (Fig. 4). As shown in Fig. 8, our model "fine-DRM" significantly performed best of all. This is because a DRM models a route from an image to the label during the training, while a DBN and DBM do not. As observed, the DRM is a bidirectional generative model, and if the parameters are specialized and tuned as a directional, discriminative model (i.e., "fine-DRM"), the performance was improved.

5.3.2 Generation Task

As we generate the label given an image using a DRM, it will be possible to generate the image given a label, because a DRM models the joint distribution of the two. To examine the potential of this possibility, we conducted image generation experiments. In these experiments, we generated images (estimated $\boldsymbol{x}$) given the one-hot labels $\boldsymbol{y}$ through mean-field updates in a similar manner to the generation scheme (reverse up and down in the left side of Fig. 2). Essentially, this procedure generates an image from $\boldsymbol{y}$; however, the dimension of 10 for the vector $\boldsymbol{y}$ is too small to estimate the upper features properly through the iterative updates. Therefore, we gave the initial values of hidden units and $\boldsymbol{x}$ for each class label as the means of the hidden units and the first visible units, respectively, calculated from the training data. After that, we obtained the images for each one-hot vector of the labels as shown in Fig. 9 (a). For comparison, we obtained images using DBM and RBM. Both models feed a concatenated vector of the image and label $[\boldsymbol{x}^\top \boldsymbol{y}^\top]^\top$ as input, and generated images in a similar procedure to the DRM. That is, we set the mean values of hidden units and the input $[\mathbb{E}[\boldsymbol{x}]^\top \boldsymbol{y}^\top]^\top$ as initial values for each class label, where $\mathbb{E}[\boldsymbol{x}]$ is expectation of $\boldsymbol{x}$ from the training data, and repeated updates of the hidden values and $\boldsymbol{x}$. The images obtained from the DBM and RBM are shown in Figs. 9 (b) and (c), respectively. For reference, images obtained from just calculating the means of each pixel for each class is shown in Fig. 9 (d). Obviously, the mean images are blurred and obscure. The images from DBM and RBM are not so blurred as the means; however, there are many pixel-wise errors to imagine the true number-images. On the other hand, the images from DRM are very clear and sharpened, and fairly

resembles real handwritten digits. This is because the proposed model can capture high-order correlations between the image and labels by considering the cyclic path between them. During the training of DRM, the estimated images from the given labels are propagated to the label layer, the estimated labels are also compared to the correct labels, and vice versa.

6. Conclusion

In this paper, we investigated our joint probability model of two kinds of visible variables, called a deep relational model (DRM), that has a hierarchical architecture to capture the latent, complicated, high-order relationships between the two, especially aimed at the improvement of classification accuracies. The DRM is viewed as one of the energy-based models, and the parameters are trainable using maximum likelihood estimation with mean-field approximation. In the image recognition experiments, we showed that the DRM with fine-tuning performed best of all the comparable deep learning models. We also showed that the DRM even without fine-tuning outperformed the discriminative DNN. In the image generation experiments, we obtained considerably realistic images from the DRM. It would be also possible to extend the proposed model so as to feed real-valued data by replacing the Bernoulli formulation regarding Eqs. (19) and (21) with the Gaussian formulation. In the future, we would like to investigate such potential when we apply it to other tasks having different kinds of representations, such as the image and speech signal, the text and speech, etc.

Acknowledgements

This work was partially supported by the Telecommunications Advancement Foundation Grants, and by JST ACT-I.

References

[1] G.E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," Neural Computation, vol.18, no.7, pp.1527–1554, 2006.
[2] D.H. Ackley, G.E. Hinton, and T.J. Sejnowski, "A learning algorithm for Boltzmann machines," Cognitive Science, vol.9, no.1, pp.147–169, 1985.
[3] P. Smolensky, "Information processing in dynamical systems: Foundations of harmony theory," Parallel Distributed Processing, vol.1, pp.194–280, 1986.
[4] V. Nair and G. Hinton, "3-D object recognition with deep belief nets," Advances in Neural Information Processing Systems, vol.22, pp.1339–1347, 2009.
[5] T. Deselaers, S. Hasan, O. Bender, and H. Ney, "A deep learning approach to machine transliteration," Proc. 4th Workshop on Statistical Machine Translation, pp.233–241, Association for Computational Linguistics, 2009.
[6] A.-r. Mohamed, G.E. Dahl, and G. Hinton, "Acoustic modeling using deep belief networks," IEEE Trans. Audio Speech Language Process., vol.20, no.1, pp.14–22, 2012.
[7] T. Nakashika, R. Takashima, T. Takiguchi, and Y. Ariki, "Voice conversion in high-order eigen space using deep belief nets," Proc. INTERSPEECH 2013, pp.369–372, 2013.
[8] R. Salakhutdinov and G.E. Hinton, "Deep Boltzmann machines," Proc. International Conference on Artificial Intelligence and Statistics, pp.448–455, 2009.
[9] R. Salakhutdinov and H. Larochelle, "Efficient learning of deep Boltzmann machines," Proc. International Conference on Artificial Intelligence and Statistics, pp.693–700, 2010.
[10] Y. Bengio, L. Yao, G. Alain, and P. Vincent, "Generalized denoising auto-encoders as generative models," Advances in Neural Information Processing Systems 26 (NIPS 2013), pp.899–907, 2013.
[11] S.M. Ali Eslami, N. Heess, C.K.I. Williams, and J. Winn, "The shape Boltzmann machine: A strong model of object shape," Int. J. Comput. Vis., vol.107, no.2, pp.155–176, 2014.
[12] H. Poon and P. Domingos, "Sum-product networks: A new deep architecture," Proc. 2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops), pp.689–690, IEEE, 2011.
[13] T. Nakashika, T. Takiguchi, and Y. Ariki, "Modeling deep bidirectional relationships for image classification and generation," 2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp.1327–1331, IEEE, 2016.
[14] B. Kosko, "Bidirectional associative memories," IEEE Trans. Syst. Man Cybern., vol.18, no.1, pp.49–60, 1988.
[15] J. Ngiam, Z. Chen, P.W. Koh, and A.Y. Ng, "Learning deep energy models," Proc. 28th International Conference on Machine Learning (ICML-11), pp.1105–1112, 2011.
[16] H. Larochelle and Y. Bengio, "Classification using discriminative restricted Boltzmann machines," Proc. 25th International Conference on Machine Learning, pp.536–543, 2008.
[17] H. Larochelle, M. Mandel, R. Pascanu, and Y. Bengio, "Learning algorithms for the classification restricted Boltzmann machine," Journal of Machine Learning Research, vol.13, pp.643–669, 2012.
[18] A. Fischer and C. Igel, "Empirical analysis of the divergence of Gibbs sampling based learning algorithms for restricted Boltzmann machines," International Conference on Artificial Neural Networks, Lecture Notes in Computer Science, vol.6354, pp.208–217, Springer, Berlin, Heidelberg, 2010.
[19] K. Cho, T. Raiko, and A.T. Ihler, "Enhanced gradient and adaptive learning rate for training restricted Boltzmann machines," Proc. 28th International Conference on Machine Learning (ICML-11), pp.105–112, 2011.
[20] G. Desjardins, A. Courville, Y. Bengio, P. Vincent, and O. Delalleau, "Parallel tempering for training of restricted Boltzmann machines," Proc. Thirteenth International Conference on Artificial Intelligence and Statistics, pp.145–152, 2010.
[21] G. Desjardins, A. Courville, and Y. Bengio, "Adaptive parallel tempering for stochastic maximum likelihood learning of RBMs," NIPS 2010 Workshop on Deep Learning and Unsupervised Feature Learning, pp.1–8, 2010.
[22] K. Cho, A. Ilin, and T. Raiko, "Improved learning of Gaussian-Bernoulli restricted Boltzmann machines," ICANN, Lecture Notes in Computer Science, vol.6791, pp.10–17, Springer, Berlin, Heidelberg, 2011.
[23] K.H. Cho, T. Raiko, and A. Ilin, "Gaussian-Bernoulli deep Boltzmann machine," Proc. International Joint Conference on Neural Networks (IJCNN), pp.1–7, IEEE, 2013.
[24] L. Deng and D. Yu, "Deep learning: Methods and applications," Foundations and Trends in Signal Processing, vol.7, no.3, pp.197–387, 2014.

Toru Nakashika received his B.E. and M.E. degrees in computer science from Kobe University in 2009 and 2011, respectively. On the summer in 2010, he was a student researcher at IBM Research, Tokyo Research Laboratory. From September 2011 to August 2012, he was a visiting researcher in the image group at INSA de Lyon in France. In the same year, he continued his research as a doctoral student at Kobe University, and received his Dr.Eng. degree in computer science in 2014. To March 2015, he was an Assistant Professor at Kobe University. He has been an Assistant Professor at the University of Electro-Communications since April 2015. His research interests include statistic signal processing, pattern recognition, and deep learning. He received the IEICE ISS Young Researcher's Award in Speech Field in 2013. He is a member of IEEE, IEICE and ASJ.