Siqi Nie, Meng Zheng, and Qiang Ji

The Deep Regression Bayesian Network and Its Applications

Probabilistic deep learning for computer vision

Deep directed generative models have attracted much attention recently due to their generative modeling nature and powerful data representation ability. In this article, we review different structures of deep directed generative models and the learning and inference algorithms associated with the structures. We focus on a specific structure that consists of layers of Bayesian networks (BNs) due to the property of capturing inherent and rich dependencies among latent variables. The major difficulty of learning and inference with deep directed models with many latent variables is the intractable inference due to the dependencies among the latent variables and the exponential number of latent variable configurations. Current solutions use variational methods, often through an auxiliary network, to approximate the posterior probability inference. In contrast, inference can also be performed directly without using any auxiliary network to maximally preserve the dependencies among the latent variables. Specifically, by exploiting the sparse representation with the latent space, max-max instead of max-sum operation can be used to overcome the exponential number of latent configurations. Furthermore, the max-max operation and augmented coordinate ascent (AugCA) are applied to both supervised and unsupervised learning as well as to various inference. Quantitative evaluations on benchmark data sets of different models are given for both data representation and feature-learning tasks.

Introduction

Deep learning has become a major enabling technology for computer vision. By exploiting its multilevel representation and the availability of big data, deep learning has led to dramatic performance improvements for certain tasks. Among different deep-learning architectures, convolutional neural networks (CNNs) have achieved the most significant development and are being widely employed in computer vision in recent years. CNNs, however, are typically discriminative models and they are built mainly for discriminative tasks, such as classification. For generative modeling, the model needs capture the underlying data distribution as well

Digital Object Identifier 10.1109/MSP.2017.2763440
Date of publication: 9 January 2018

as the mechanisms used to generate data, including the uncertainties in the data and in the data generation process. CNNs are therefore not directly suitable for generative modeling. The latest development in generative adversarial networks (GANs) [8] can perform effective data generation. Based on combining a discriminative CNN with a generative deconvolutional neural network, GANs produce remarkable performance in generating realistic data. Since GANs employ a simple standardized random vector to model data uncertainty; they remain mostly deterministic and cannot fully model the uncertainties in data.

The major difficulty of learning and inference with deep directed models with many latent variables is the intractable inference due to the dependencies among the latent variables and the exponential number of latent variable configurations.

In this article, we focus on fully probabilistic deep directed generative models for deep learning under uncertainty that can simultaneously perform data generation and classification tasks. Based on probability theories, deep probabilistic generative models offer a probabilistically grounded framework to represent, learn, and predict under uncertainty. There exist several types of probabilistic deep generative models, most notably, deep Boltzmann machines (DBMs) [29] [Figure 1(c)] and deep belief networks (DBNs) [11] [Figure 1(b)]. While generative in nature, these models, for the sake of simplicity in inference, typically assume latent variables are independently given data. Such an assumption weakens their data modeling and representation power. In contrast, deep directed generative models consist of layers of BNs as shown in [Figure 1(a)]. Compared to DBMs and DBNs, the deep directed generative model enjoys several advantages due to its unique way of capturing dependencies. First, it specifically models the data generation process and allows straightforward ancestry sampling, where a node is sampled following a topological order, starting from its ancestors until its descendants. Second, there is no intractable partition function that has plagued the undirected models. Last, but most importantly, latent nodes in the deep directed generative models are dependent on each other given inputs (because of the "explaining away" principle) while it is not the case for DBN and DBM. This characteristic of the deep directed generative models endows them a powerful ability in modeling the complex pattern in data.

Learning and inference in deep directed generative models is challenging, mainly due to the intractable computation of the posterior probability of the latent variables and the exponential number of latent configurations (for binary latent variables). Various approximations such as variational inference algorithms [2], [8], [10], [15], [21], [26] have been proposed to address these challenges. They all use an auxiliary feed-forward network to approximately solve the intractable posterior probability inference problem. These approximations typically assume the joint latent variable distribution given data can be factorized. The factorized distribution, while simplifying learning and inference, sacrifices the dependencies among the latent variables for efficiency. They inevitably enlarge the distance to the true posterior and weaken the representation power of the model. This negates a major advantage of the directed graphical models. Moreover, the existing methods avoid the exponential number of latent configurations, as they mostly deal with real latent nodes.

Alternatively, the posterior inference can be solved directly, without resorting to any other network. Furthermore, efficient learning and inference methods can be designed to maximally preserve the dependencies among the latent variables. Specifically, for learning, data marginal log-likelihood can be maximized directly through a max-max operation to overcome the exponential number of latent configurations. For inference, posterior probability inference can be performed through the pseudo-likelihood, which also preserves dependencies of latent variables. Furthermore, improved maximum a posteriori (MAP) inference can be achieved by combining the coordinate ascent (CA) method with the variational method.

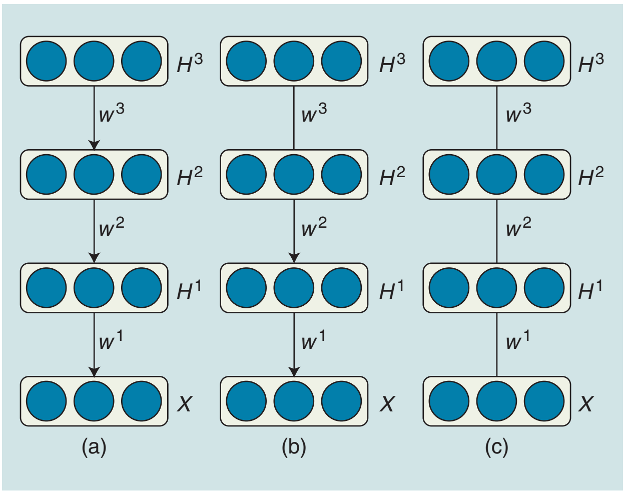

FIGURE 1. Graph representation of (a) DRBNs, (b) DBNs, and (c) DBMs, where arrows represent directed links and line segments represent undirected links. Directed links capture the causal or one-way dependency between the connected nodes, while the undirected links capture the mutual dependencies among the connected nodes.

Related work

In general, there are three types of deep probabilistic generative models: fully directed, hybrid, and fully undirected. The DBN [Figure 1(b)] has a hybrid structure, where the top two layers are connected with undirected links and all other layers are connected with directed links. The DBM [Figure 1(c)] has a fully undirected structure such that every two consecutive layers are connected using undirected links. In this article, we focus on the fully directed deep generative models due to their unique representation power. Depending on the structure and the type of variables, the directed generative models have several variants. The most commonly used structure consists of multiple layers [Figure 1(a)], in which only the bottom layer represents the visible variables. The connections are directed from the upper layer to the lower layer, and no connection within each layer is allowed.

Depending on the types of variables, deep directed generative models can be categorized into deep sigmoid belief networks (SBNs) [21], [24], [31], deep Poisson factor analysis (DPFA) [6], deep factor analyzers (DFAs) [34], and deep Gaussian mixture models (DGMMs) [36]. The deep SBN contains binary latent and visible variables, and the conditional probability is defined using a sigmoid function. The DPFA models discrete variables (e.g., word count in documents) using binary latent variables. Dirichlet prior ϕ and gamma prior θ are placed to describe a Poisson distribution of input data. The DFA consists of continuous latent and mixtured types of variables. The DGMM is an extension of the Gaussian mixture model (GMM), while each latent node represents a linear operation to compute the mean and covariance matrix for the Gaussian distribution.

Compared to DBMs and DBNs, the deep directed generative model enjoys several advantages due to its unique way of capturing dependencies.

In this article, we focus on the deep directed generative model with binary latent variables. Compared to earlier SBN works [21], [24], [31], the deep directed generative models represent an extension in both representation and learning and inference methods. In terms of representation, it allows for different input and output types (discrete, continuous, and hybrid), while SBN only uses binary input. For learning, DRBN allows both unsupervised and supervised layer-wise and global learning, while SBN work only includes layer-wise unsupervised learning. For inference, DRBN allows the use of different algorithms such as the pseudo-likelihood method for posterior probability inference and AugCA method for MAP inference, while the literature for SBNs generally use variational methods for inference.

For deep directed generative models, computing the posterior becomes intractable due to dependencies among the latent variables. To address this issue, one approach is to design a special prior to make the latent variables conditionally independent such as the complementary prior [11] for DBNs, wherein the posterior probability for each latent variable can be computed individually. Another popular approach is to replace the true posterior distribution with a factorized distribution as an approximation, known as variational methods. The mean field theory [31] for learning SBNs makes the latent variables totally independent. A set of variational parameters is learned to minimize the Kullback–Leibler (KL) divergence between the true posterior and the approximate one. Another approximate inference algorithm is the Markov chain Monte Carlo method, which can be used to estimate the posterior probability of both latent variables and parameters. One example is the learning and inference for deep latent Gaussian models [14].

To extend the traditional variational methods, recent works typically use an auxiliary network to address the computational challenge with posterior probability inference. Specifically, the wake-sleep algorithm [13] augments the SBNs with a feed-forward recognition network for efficient inference. Mnih and Gregor [21] propose a variational inference network with discrete latent variables to perform efficient inference. Rezende et al. [26] and Kingma and Welling [15] introduced a recognition model to efficiently approximate posterior probability inference. The deep generative model can be learned by jointly optimization of the parameters for both generative and the recognition models. Ranganath et al. [25] introduces a new method to reduce the variance of the noisy gradients for varational based deep model learning. The overall structure of [2] is similar to [15], by building connections between the recognition model and generative model rather than learning them independently.

All of these approaches use a feed-forward model to perform posterior probability inference. The feed-forward models all assume the posterior probability can be factorized. The posterior probability inference is, hence, approximate. In contrast, posterior probability inference can be done via CA without the help of any auxiliary network or variational distribution, therefore, during learning, we do not need to assume conditional independence among the latent variables. Moreover, by initializing the CA method with the inference result from a variational method, the AugCA method can produce better reconstruction results than the existing variational models.

One of the latest development in deep generative modeling is the GAN [8], [10], which employs both a discriminator and a generator. The generator is a deep generative model, which is trained to generate realistic synthetic data. The discriminator is a neural network trained to discriminate synthetic data from real data. The two networks compete against with each other until the discriminator is maximally confused. Specifically, Han et al. [10] uses a nonlinear factor analyzer as the generator network. By performing the alternating backpropagation, the generator network can be learned to generate realistic images, sequences, and sounds. Despite its impressive generative performance, GANs are essentially a deterministic model as the data uncertainty is modeled by a standard random vector. Hence, they cannot effectively capture the probabilistic distribution of the data. Furthermore, GANs can only perform data generation tasks and cannot perform classification tasks. By combining a variational autoencoder with a GAN, Larson et al. [16] introduced a novel metric for similarity measurement when reconstructing the training samples during parameter updating, which achieves better generalization performance. Variational autoencoders and GANs are further combined in [20], with a clear theoretical justification and the ability for arbitrary complex inference.

Deep regression BNs

Regression BNs

To construct a deep directed generative model, a BN is used as the building block. A BN is parameterized by the conditional probability for each node, given its parents. The number of parameters for each node increases exponentially with the number of parents of each node. To scale up to models with a large number of latent variables, regression BNs (RBNs) are employed to limit the number of parameters linearly with the number of connections [27].

The RBN consists of two layers: one visible layer $\boldsymbol{X}$ of dimension n_d and one latent layer $\boldsymbol{H}$ of dimension n_h, as shown in Figure 2(a). Every latent variable connects to every visible variable with a directed edge. For continuous data vector $\boldsymbol{X} = \{X_1,...,X_{n_d}\} \in \mathbb{R}^{n_d}$ and binary latent variables $\boldsymbol{H} = \{H_1,...,H_{n_h}\} \in \{0,1\}^{n_h}$ in which each node takes value 0 or 1, the prior and conditional probabilities of the model are defined as

$$P(H_j = 1) = \mathrm{sigm}(d_j), \tag{1}$$

$$P(X_i = x_i \mid \boldsymbol{H} = \boldsymbol{h}) \sim \mathcal{N}(x_i \mid \boldsymbol{w}_i^T \boldsymbol{h} + b_i, \sigma_i^2), \tag{2}$$

where $\mathrm{sigm}(a) = 1/(1+\exp(-a))$ is the sigmoid function; $\mathcal{N}(a \mid \mu, \sigma^2)$ represents the Gaussian distribution with mean μ and variance σ^2. $\boldsymbol{W} = [\boldsymbol{w}_1,...,\boldsymbol{w}_{n_d}]^T$ is the weight matrix, w_{ij} is the weight of the link connecting nodes H_j and X_i, where $i = \{1,2,...,n_d\}$, $j = \{1,2,...,n_h\}$. $\boldsymbol{d} = [d_1,...,d_{n_h}]^T$ and $\boldsymbol{b} = [b_1,...,b_{n_d}]^T$ are the bias parameters for $\boldsymbol{H}$ and $\boldsymbol{X}$ respectively, and $\boldsymbol{\sigma^2} = [\sigma_1^2,...,\sigma_{n_d}^2]^T$ are the conditional variances for $\boldsymbol{X}$. This generative model can be viewed as a diagonal GMM with the number of components exponential in the number of latent variables. For binary data, the prior probability of the latent node remains the same and the conditional probability is

$$P(X_i = 1 \mid \boldsymbol{H} = \boldsymbol{h}) = \mathrm{sigm}(\boldsymbol{w}_i^T \boldsymbol{h} + b_i). \tag{3}$$

Through the parameterization in (2), the mean of a Bernoulli node is the sigmoid function of linear combination of all nodes in the previous layer. The RBN can be seen as a probabilistic generalization to the neural network, where the value of nodes in the current layer is the nonlinear transformation (e.g., via a sigmoid activation) of the linear combination of all nodes in the previous layer. With the prior and conditional distributions, the joint distribution for $\boldsymbol{X}$ and $\boldsymbol{H}$ is

$$P(\boldsymbol{x},\boldsymbol{h} \mid \Theta) = \prod_{j=1}^{n_h} P(h_j \mid \Theta_j) \prod_{i=1}^{n_d} P(x_i \mid \boldsymbol{h}, \Theta_i), \tag{4}$$

where Θ_i and Θ_j denote the parameters for the ith visible node and jth latent node. $\Theta = \{\boldsymbol{W}, \boldsymbol{\sigma}, \boldsymbol{b}, \boldsymbol{d}\}$. Plugging the parameterization in (1) and (2) into (4) yields

$$P(\boldsymbol{x},\boldsymbol{h} \mid \Theta) = \frac{\exp(-E(\boldsymbol{x},\boldsymbol{h} \mid \Theta))}{(2\pi)^{n_d/2} \prod_i \sigma_i \prod_j (1+\exp(d_j))}, \tag{5}$$

where

$$E(\boldsymbol{x},\boldsymbol{h} \mid \Theta) = \sum_i \frac{(x_i - b_i)^2}{2\sigma_i^2} - \sum_i \frac{x_i - b_i}{\sigma_i^2} \boldsymbol{w}_i^T \boldsymbol{h} - \boldsymbol{d}^T \boldsymbol{h} + \sum_i \frac{1}{2\sigma_i^2} (\boldsymbol{w}_i^T \boldsymbol{h})^2. \tag{6}$$

Similarly, we can produce the joint probability for the RBN with binary input by combining (3) with (1). The energy function in (6) is very similar to the energy function for the Gaussian–Bernoulli restricted Boltzmann machine (GRBM) [12] in (7).

$$E_{\mathrm{RBM}}(\boldsymbol{x},\boldsymbol{h}) = \sum_i \frac{(x_i - b_i)^2}{2\sigma_i^2} - \sum_i \frac{x_i}{\sigma_i^2} \boldsymbol{w}_i \boldsymbol{h} - \boldsymbol{d}^T \boldsymbol{h}. \tag{7}$$

Comparing the two equations, it is clear that the energy function for RBN has an extra term [the last term of (6)]. This extra term explicitly captures the relationship among latent variables. This represents a major representation difference between RBNs and RBMs.

Deep RBNs

By stacking multiple RBNs, we are able to construct a deep directed generative model with L latent layers, called *deep RBNs* (*DRBNs*) [Figure 2(b)]. Let $\boldsymbol{H}^l, l = 1,\ldots,L$ denote the binary latent variables in layer l, and $\boldsymbol{H}^0 = \boldsymbol{X}$. Denote the parameters between layer $\boldsymbol{H}^{l-1}$ and $\boldsymbol{H}^l$ as $\Theta^l = \{\boldsymbol{W}^l, \boldsymbol{b}^l\}$. Top layer parameters are d. The prior probability for a variable of the top layer is

$$P(H_j^L = 1) = \mathrm{sigm}(d_j). \tag{8}$$

The conditional probability for the remaining layers is

$$P(H_j^{l-1} = 1 \mid \boldsymbol{H}^l = h^l) = \mathrm{sigm}(\boldsymbol{w}_j^{lT} \boldsymbol{h}^l + b_j^l), \quad 2 \le l \le L. \tag{9}$$

The conditional probability $P(\boldsymbol{x} \mid \boldsymbol{h}^1)$ for the bottom two layers is the same as in (2) or (3), depending on the

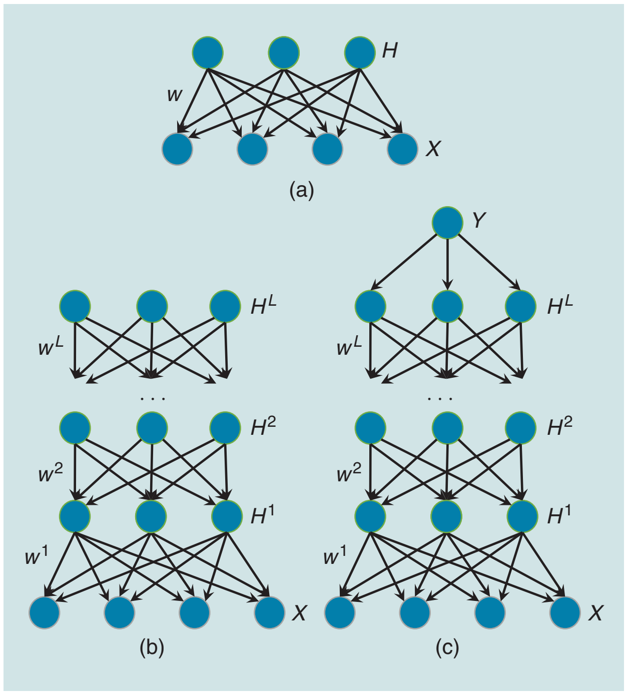

FIGURE 2. A graph representation of (a) an RBN, (b) a DRBN without labels, and (c) a DRBN with labels.

type of the input data. The joint probability for all variables is

$$P(\boldsymbol{x},\boldsymbol{h}^1,\ldots,\boldsymbol{h}^L \mid \Theta) = P(\boldsymbol{x} \mid \boldsymbol{h}^1)P(\boldsymbol{h}^L)\prod_{l=2}^{L} P(\boldsymbol{h}^{l-1} \mid \boldsymbol{h}^l). \quad (10)$$

By fully capturing the joint probability distribution of input data and latent variables, the DRBN can accurately capture three types of dependencies among different variables: dependencies among input variables, dependencies among hidden variables, and interactions between hidden and input variables. Because of its explicit capture of these dependencies, compared with DBNs and DBMs, DRBNs can better capture the data distribution. Compared with the GANs, DRBNs explicitly capture the underlying probabilistic data distribution and can perform both data modeling and prediction tasks.

Comparison with deep neural networks

Compared with deep neural networks (DNNs), DRBNs consist of layers of RBNs. Each hidden layer of the BN hence captures the distribution of the input data at different levels. In contrast, DNNs consist of layers of perceptions and each hidden layer summarizes the sufficient statistics (such as the mean) of the input at different levels. Therefore, the main difference between DRBNs and DNNs is that the DRBN captures the probabilistic distribution of the data, while the DNN captures mean statistics of the data. Based on this understanding, DRBNs represent a generalization to the DNN, and the DRBN becomes a DNN if the conditional variance for each latent node becomes zero. The DRBN is therefore more powerful in data representation, in particular in representing the uncertainties in the data. On the other hand, the power in representation also leads to challenges in DRBN learning and inference and in its ability to scale up. DNNs are much better than DRBNs in efficient inference and learning, and in scaling up to a large model. This explains why, so far, DNNs remain the dominant deep model architecture. But the promise of DRBNs in data representation deserves further research to address its challenges.

The promise of DRBNs in data representation deserves further research to address its challenges.

DRBN learning

Unsupervised learning

We first discuss the unsupervised RBN learning and then extend it to DRBN learning. The goal of unsupervised parameter learning for an RBN is to estimate the parameters Θ given a set of data samples $\mathcal{D} = \{\boldsymbol{x}^m\}_{m=1}^{M}$. To maximally preserve the dependencies among the latent variables,it would be better to directly maximize the marginal log-likelihood instead of its surrogate (such as its lower bound), i.e.,

$$\begin{aligned}\Theta^* &= \operatorname*{argmax}_{\Theta} \log P(\mathcal{D} \mid \Theta) \\ &= \operatorname*{argmax}_{\Theta} \sum_m \log \sum_{\boldsymbol{h}} P(\boldsymbol{x}^m, h \mid \Theta). \end{aligned} \quad (11)$$

Maximizing the marginal likelihood has two computational challenges: 1) computing the posterior probability $P(\boldsymbol{h} \mid \boldsymbol{x})$ is intractable even for one configuration $\boldsymbol{h}$ due to the dependencies among elements of $\boldsymbol{h}$ and 2) there are exponential number of terms in the summation over $\boldsymbol{h}$. To address these two challenges, Θ can be estimated by replacing the max-sum operation in (11) with the max-max operation in (12), i.e.,

$$\begin{aligned}\Theta^* &= \operatorname*{argmax}_{\Theta} \log P(\mathcal{D} \mid \Theta) \\ &\approx \operatorname*{argmax}_{\Theta} \sum_m \log \max_{\boldsymbol{h}} P(\boldsymbol{x}^m, \boldsymbol{h} \mid \Theta). \end{aligned} \quad (12)$$

The rational for replacing the max-sum with max-max operation is based on the empirical observation that the distribution of $P(\boldsymbol{h} \mid \boldsymbol{x}^m)$ is very sparse, with its energy concentrated on a few configurations of $\boldsymbol{h}$ near its maximum. The max-max operation may be achieved iteratively in two steps. First, for each training sample $\boldsymbol{x}^m$, we obtain $\boldsymbol{h}^{*m}$ that maximizes $P(\boldsymbol{h},\boldsymbol{x}^m \mid \Theta_{t-1})$, given current parameters Θ_{t-1}, i.e.,

$$\boldsymbol{h}^{*m} = \operatorname*{argmax}_{\boldsymbol{h}} P(\boldsymbol{h},\boldsymbol{x}^m \mid \Theta_{t-1}). \quad (13)$$

Second, given $\boldsymbol{h}^{*m}$, Θ_t can be estimated by maximizing $\sum_m \log P(\boldsymbol{x},\boldsymbol{h}^*_m \mid \Theta_t)$, i.e.,

$$\Theta_t^* = \operatorname*{argmax}_{\Theta} \log \sum_m P(\boldsymbol{x}^m, \boldsymbol{h}^{*m} \mid \Theta_t). \quad (14)$$

The two steps iterate until convergence. Equation (13) can be solved through the AugCA method to be discussed in the section "MAP Inference." Equation (14) can be solved in closed form solution for continuous input or through stochastic gradient ascent for binary input.

Learning a DRBN with multiple latent layers consists of two steps: layer-wise pretraining and global fine-tuning. Layer-wise pretraining is a bottom-up procedure. When learning the parameters Θ^l for the lth layer, the parameters below are frozen, and the input to the lth layer is H^{*l}, which is obtained though $\boldsymbol{H}^{*l} = \operatorname{argmax}_{\boldsymbol{H}^l} P(\boldsymbol{H}^l \mid \boldsymbol{H}^{l-1}, \Theta^{l-1})$. Given its input, Θ^l can be learned the same way using the RBN learning method. The layer-wise learning does not consider the interactions among the layers. A global fine-tuning can then be performed. Initialized by the parameters learned by layer-wise learning, the global fine-tuning procedure simultaneously refines the parameters at different layers such that parameters in the higher layers can influence those in the lower layers. Specifically, global fine-tuning updates the parameters $\boldsymbol{\Theta}$ in all layers by maximizing the marginal likelihood of the data, i.e.,

$$\begin{aligned}\Theta^* &= \arg\max_{\Theta} \sum_m \log P(\boldsymbol{x}^m \mid \boldsymbol{\Theta}) \\ &\approx \arg\max_{\Theta} \sum_m \log \max_{\boldsymbol{h}^1,\ldots,\boldsymbol{h}^L} P(\boldsymbol{x}^m, \boldsymbol{h}^1, \ldots \boldsymbol{h}^L \mid \boldsymbol{\Theta}). \end{aligned}$$

Equation (15) can be solved through two step iteration as for (12).

Supervised learning

For applications where labels $\boldsymbol{y}^m$ are given for each sample $\boldsymbol{x}^m$ as shown in Figure 1, supervised learning can be performed. For classification tasks,discriminative supervised learning can be performed. The objective function is modified to maximize the log posterior probability of the labels,

$$\begin{aligned}\boldsymbol{\Theta}^* &= \underset{\Theta}{\operatorname{argmax}} \log P(\mathcal{Y} \mid \mathcal{D}, \boldsymbol{\Theta}) \\ &= \underset{\Theta}{\operatorname{argmax}} \sum_m \log P(y^m \mid \boldsymbol{x}^m, \boldsymbol{\Theta}) \\ &= \underset{\Theta}{\operatorname{argmax}} \sum_m \log \sum_{\boldsymbol{h}} P(y^m, \boldsymbol{h} \mid \boldsymbol{x}^m, \boldsymbol{\Theta}). \qquad (15)\end{aligned}$$

We encounter the same computational challenges in solving (15). This challenge can be alleviated by replacing the max-sum operation with the max-max operation (16), i.e., which can be solved similarly through the two-step iteration process

$$\Theta^* = \underset{\Theta}{\operatorname{argmax}} \sum_m \log \max_{\boldsymbol{h}} P(y^m, \boldsymbol{h} \mid \boldsymbol{x}^m, \Theta). \qquad (16)$$

Like unsupervised learning, (16) can be applied to both layer-wise and global DRBN learning. To achieve a good initialization of parameters, the generative model is first trained in an unsupervised manner. The objective function is then changed to the supervised one, and parameters are tuned in a supervised manner.

DRBN inference

Given a DRBN with known parameters, there are three types of inferences: posterior probability inference, MAP inference, and likelihood inference. In this section, we discuss efficient methods for these three types of inferences.

Posterior probability inference

The posterior inference is to compute the posterior probability of the latent variables given the input data, i.e., to compute $P(\boldsymbol{h} \mid \boldsymbol{x})$. Because of the dependencies among $\boldsymbol{h}$, directly computing $P(\boldsymbol{h} \mid \boldsymbol{x}, \Theta)$ is computationally intractable when the dimension of $\boldsymbol{h}$ is high. The pseudo-likelihood method offers an efficient solution to this problem by replacing the conditional likelihood with a more tractable objective. The pseudo-likelihood method considers the following approximation:

$$P(\boldsymbol{h} \mid \boldsymbol{x}, \Theta) \approx \prod_j P(h_j \mid \boldsymbol{h}_{-j}, \boldsymbol{x}, \Theta), \qquad (17)$$

where $\boldsymbol{h}_{-j} = \{h_1, \ldots, h_{j-1}, h_{j+1}, \ldots, h_{n_h}\}$. In this approximation, conditioning is added over additional variables. The conditional pseudo-likelihood can be factorized into local conditional probabilities, which can be computed in parallel. The pseudo-likelihood approximation, however, requires an initialization of $\boldsymbol{h}$.

MAP inference

MAP inference is to estimate the most likely hidden layer configuration $\boldsymbol{h}^*$, given an input $\boldsymbol{x}$, i.e., $\boldsymbol{h}^* = \operatorname{argmax}_{\boldsymbol{h}} P(\boldsymbol{h} \mid \boldsymbol{x})$. Directly performing MAP inference is computationally intractable as we need enumerate all possible hidden layer configurations. The CA method was introduced to overcome this challenge, by iteratively maximizing one latent variable at a time. From an initial state of the latent vector $\boldsymbol{h}^{(0)}$, the CA method updates one latent variable by fixing all other variables iteratively,

$$h_j^{(t+1)} = \underset{h_j}{\operatorname{argmax}} P(h_j \mid \boldsymbol{x}, \boldsymbol{h}_{-j}^{(t)}). \qquad (18)$$

This iterative updating rule guarantees that the posterior probability $P(\boldsymbol{h} \mid \boldsymbol{x})$ will only increase until convergence due to the inequality: $P(h_j^{t+1}, \boldsymbol{h}_{-j}^t \mid \boldsymbol{x}) \geq P(\boldsymbol{h}^t \mid \boldsymbol{x})$. As a greedy approach, the CA method may get stuck in a local optimum. Thus, the initialization for (18) is crucial to ensure a configuration with high quality. To address this issue, the variational inference approach may be employed to learn an inference network [21] from the DRBN. The inference result from the inference network is used as initialization for the CA, yielding the AugCA method.

The inference network method [21] approximates the posterior using only one set of parameters for all data samples by defining a feed-forward network for $Q(\boldsymbol{h} \mid \boldsymbol{x}, \phi)$. The inference network also assumes independencies among latent variables given input data

$$Q(\boldsymbol{h} \mid \boldsymbol{x}, \phi) = \prod_j q(h_j \mid \boldsymbol{x}, \phi_j). \qquad (19)$$

Each individual probability is defined using a sigmoid function,

$$q(h_j \mid \boldsymbol{x}, \phi_j) = \operatorname{sigm}\Big(\sum_i v_{ij} \boldsymbol{x}_i + s_j\Big). \qquad (20)$$

The parameters $\phi = \{\boldsymbol{v}, \boldsymbol{s}\}$ are learned by minimizing the average KL divergence $KL(Q(\boldsymbol{h} \mid \boldsymbol{x}, \phi) \| P(\boldsymbol{h} \mid \boldsymbol{x}, \Theta))$ over all of the training samples. Similarly, the optimization is through the expected log-likelihood due to the intractable $P(\boldsymbol{h} \mid \boldsymbol{x}, \Theta)$,

$$\phi^* = \underset{\phi}{\operatorname{argmax}} \sum_{\boldsymbol{x}} \sum_{\boldsymbol{h}} Q(\boldsymbol{h} \mid \boldsymbol{x}, \phi) \log \frac{P(\boldsymbol{h}, \boldsymbol{x} \mid \Theta)}{Q(\boldsymbol{h} \mid \boldsymbol{x}, \phi)}. \qquad (21)$$

The CA method requires the computation of $P(h_j^t = 1 \mid \boldsymbol{x}, \boldsymbol{h}_{-j}^{t-1})$ for each j. Naive computation can become prohibitive, given a large number of hidden variables. Taking discrete input data as illustration, according to (4), the computation of joint probability requires $N_h + N_d$ times of multiplication of the exponential terms. This can become very expensive when N_d and N_d are large, since each time, only one element of $\boldsymbol{h}$ is changed and the probabilities for other elements remain the same. An efficient procedure to recursively compute $P(h_j^t = 1 \mid \boldsymbol{x}, \boldsymbol{h}_{-j}^{t-1})$ can be written as

$$\begin{aligned}&P(h_j^t = 1 \mid \boldsymbol{x}, \boldsymbol{h}_{-j}^{t-1}) \\ &\quad = \frac{P(h_j^t = 1, \boldsymbol{x}, \boldsymbol{h}_{-j}^{t-1})}{P(h_j^t = 1, \boldsymbol{x}, \boldsymbol{h}_{-j}^{t-1}) + P(h_j^t = 0, \boldsymbol{x}, \boldsymbol{h}_{-j}^{t-1})} \\ &\quad = \frac{1}{1 + \dfrac{P(h_j^t = 1, \boldsymbol{x}, \boldsymbol{h}_{-j}^{t-1})}{P(h_j^t = 0, \boldsymbol{x}, \boldsymbol{h}_{-j}^{t-1})}}, \qquad (22)\end{aligned}$$

where the probability ratio can be further written as

$$\frac{P(h_j^t=1,\boldsymbol{x},\boldsymbol{h}_{-j}^{t-1})}{P(h_j^t=0,\boldsymbol{x},\boldsymbol{h}_{-j}^{t-1})}=\exp(d_j)\exp(x_i\boldsymbol{w}_j)\cdot \prod_i \frac{1+\exp\left(\sum_{n=1,n\neq j}^{n_h} w_{i,n}h_n^{t-1}+w_{i,j}+b_i\right)}{1+\exp\left(\sum_{n=1,n\neq j}^{n_h} w_{i,n}h_n^{t-1}+b_i\right)}. \tag{23}$$

The first two terms in (23) are constant for a given $\boldsymbol{x}$. They need be computed only once. For the last term, either the numerator or denominator can be retrieved from the last iteration, depending on the last value of h_j^{t-1}. This means we only need compute either the numerator or denominator in each iteration. This cuts the computational time by half compared to the naive computation. Furthermore, the term within the exponential term can be retrieved and updated from last iteration by one addition or subtraction. Then the probability ratio in (23) can be obtained by n_d times of multiplication. As a result, computation time for each iteration is constant, involving one addition or subtraction and n_d times of multiplication. The overall complexity of the CA method is $O(n_d n_h n_i)$, which is linear in the size of the input data, the number of latent variables n_h, and the number of iterations n_i.

Likelihood inference

The likelihood inference is to compute the marginal probability of the visible variables, i.e., $P(\boldsymbol{x})$. Since

$$P(\boldsymbol{x})=\sum_{\boldsymbol{h}} P(\boldsymbol{h}\mid\boldsymbol{x})P(x). \tag{24}$$

By replacing the sum operation with the max operation, we have

$$P(\boldsymbol{x})\approx \max_{\boldsymbol{h}} P(\boldsymbol{h}\mid\boldsymbol{x})P(\boldsymbol{x})=\max_{\boldsymbol{h}} P(\boldsymbol{h},\boldsymbol{x}). \tag{25}$$

It again can be computed by first computing $\boldsymbol{h}^*=\text{argmax}_{\boldsymbol{h}}P(\boldsymbol{h}\mid\boldsymbol{x})$ using the AugCA method. This can then be followed by easily computing $P(\boldsymbol{x},\boldsymbol{h}^*)$.

Algorithm evaluation

As discussed in the section "Related Work," the DRBN's learning method is different from variational methods, which use a factorized variational distribution to approximate the true posterior probability inference. In contrast, the DRBN directly maximizes the marginal log-likelihood through a max-max operation To demonstrate the advantage of such a learning algorithm, we performed experiments to compare three learning methods: max-max learning, variational learning [21], and the exact learning method in terms of their ability in data representation under different number of latent nodes.

For this experiment, we trained the RBN networks on 60,000 training samples of MNIST data set [17] (binarized according to [23]). During learning, we respectively use the max-max operation, the variational method [21], and the exact method (only for a small network) to approximate the marginal log-likelihood. For a fair comparison, the hyperparameters for each method are optimally tuned.

Table 1. A comparison of training data log-likelihood (a small network with five hidden nodes).

Variational [21]	Max-Max	Exact
–206.0620	–170.7570	–170.5416

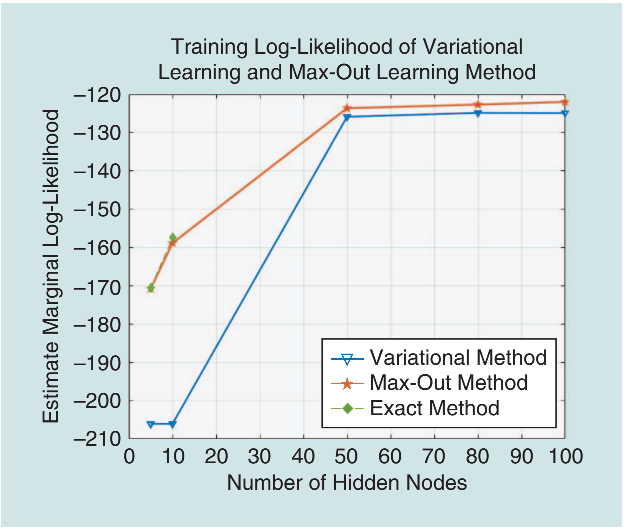

FIGURE 3. The average log-likelihood for the RBN learned with different learning methods under a different number of hidden nodes. Red line: max-out learning, blue line: variational learning, and green line: exact learning.

After learning, the mean log-likelihood of all training samples for RBNs learned with the three methods are summarized in Table 1. It is clear from Table 1 that the max-max method produces an average log-likelihood very close to the exact method and is much better than the variational method. This demonstrates the improved accuracy of the max-max method in data representation over the variational methods for a small network.

We further evaluate these methods for larger networks. For a network with a large number of hidden nodes, the exact method cannot be completed as the summation over $\boldsymbol{h}$ becomes computationally intractable. Hence, we only give the mean data log-likelihood in Figure 3 for the max-max method and the variational method under a different number of hidden nodes. The estimate mean data log-likelihood is computed the same way as in Table 1. We also include the mean log-likelihood of the exact learning method for small RBNs with five and ten hidden nodes. From the curves in Figure 3, we can see that the data log-likelihood evaluated on RBN learned with the max-max method is consistently higher than the variational method, especially when the network is small. This not only demonstrates further the advantage of the max-max learning method over the variational learning method but also proves the validity of replacing max-sum operation by the max-max operation.

Table 2. PSNRs for different image inpainting methods on Berkeley data set.

Method	PSNR
KSVD [5]	24.13
FoE [28]	24.79
GMM [40]	25.71
DRBN (CA)	29.42
DRBN (AugCA)	33.46

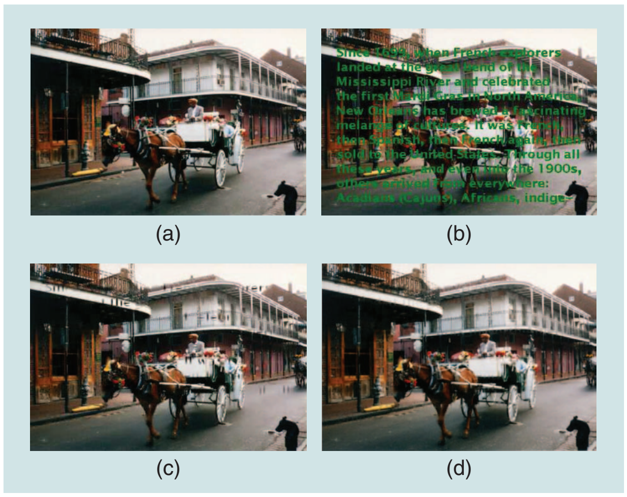

FIGURE 4. An example of the image inpainting experiment, (a) original image, (b) corrupted image, (c) restored image using the GMM, and (d) restored image using the DRBN. (Images used with permission from [19].)

Applications

To demonstrate the effectiveness of the DRBN models, we will apply the DRBN to different computer vision tasks to demonstrate its capability for both data representation and feature learning for classification.

Data representation

First, we quantitatively compare the DRBN with other generative models in terms of peak signal-to-noise ratio (PSNR) in several image restoration and denoising tasks. Then, we evaluate the generative representation power of different models in terms of generating synthetic data.

The first task is image restoration, which is to restore an image from its corrupted version. It is shown that a higher likelihood of patches leads to better denoising on whole images [40]. Therefore, we train a DRBN model on the 8×8 patches from the Berkeley data set [19], which contains 200 training and 100 testing images. One million training and 50,000 testing patches are randomly sampled from 200 training and 100 test images, respectively. A DRBN with two latent layers is used, with each layer containing 50 latent variables respectively. The total training epochs are 100. With a learned DRBN as a prior model, we use the expected patch log likelihood (EPLL) [40] framework to perform image restoration. The EPLL of an image $\boldsymbol{x}$ is defined as

$$\mathrm{EPLL}_P(\boldsymbol{x}) = \sum_i \log P(\boldsymbol{y}_i), \tag{26}$$

where $\{\boldsymbol{y}_i\}$ represents all the overlapping patches in the image. Given a corrupted image $\tilde{\boldsymbol{x}}$, the cost we use to reconstruct the image with patch prior P is

$$f_P(\boldsymbol{x} \mid \tilde{\boldsymbol{x}}) = \frac{\lambda}{2} \| \boldsymbol{x} - \tilde{\boldsymbol{x}} \|^2 - \mathrm{EPLL}_P(\boldsymbol{x}). \tag{27}$$

It is difficult to directly optimize the complicated cost function. We employ the half-quadratic slitting method [7] with λ set to 10^6, following the settings in [40]. To perform image inpainting, we superimposed some sentences on the clean image as the noise. During optimization, both the CA and AugCA methods were used to perform MAP inference for each patch. Typically the CA and AugCA converge after three iterations, and initialization affects the final configuration of latent variables but not posterior likelihood. We compared DRBN to three state-of-the-art approaches with generic priors: field of experts (FoE) [28], KSVDG [5], and the GMM [40] with full covariance matrix. The quantitative results on the 100 test images are given in Table 2. The DRBN outperforms all other approaches in terms of PSNR values.

An example is given in Figure 4, where (a) represents an original image, (b) a corrupted image, (c) a restored image using the GMM as a prior model, and (d) a restored image using the DRBN as a prior model. The PSNR values for (c) and (d) are 26.31 and 29.80, respectively. It can be seen that using the GMM prior, some parts of the letters in the corrupted image remain in the restored image, especially if the background color is white. The DRBN model completely removes the letters, and the PSNR values show significant improvement.

The second task for image representation is face restoration, where we use DRBN to restore a given cropped image. For this task, we put two kinds of noise on the face images from the Multi-PIE data set: random noise and block occlusion, where 40% of the pixels are corrupted by random noise with a standard deviation of 0.4, following the same procedure as [35]. For the latter, 12×12 blocks are superimposed on a random part of the 32×32 faces with Gaussian random noise, following the same procedure as [35]. The same MAP inference methods—CA and AugCA—are applied. Also, results from the inference network (IN) [21] is compared with CA and AugCA. The baseline methods include the robust Boltzmann machine (RoBM) [35], GRBM [12], and robust PCA (RPCA) [38]. PSNR is employed to quantitatively evaluate different methods. The result is given in Table 3. The AugCA method outperforms all other methods in terms of PSNR values. In the cross-subject setting, the RBN generalizes well to unseen subjects. Some examples of the reconstructed faces are given in Figure 5.

Next we demonstrate the DRBN's image reconstruction ability. In the image restoration task presented in Figure 5, images are restored using trained DRBN models from given corrupted images. Unlike the image reconstruction task, given an image $\boldsymbol{x}$, we first perform MAP inference to obtain the most likely latent

Table 3. PSNRs for different denoising methods on Multi-PIE data set.

Method	Random Noise	Block Occlusion
RPCA	22.53	20.14
GRBM	23.18	21.45
RoBM	27.15	24.32
RBN (CA)	28.60	27.21
RBN (IN) [21]	27.89	26.21
RBN (AugCA)	29.23	27.45

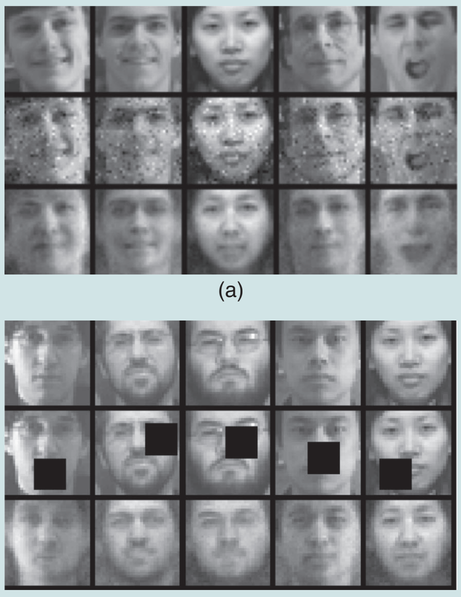

FIGURE 5. Examples of face restoration from (a) random noise and (b) block occlusion. Top rows: original images; middle rows: corrupted images; and bottom rows: reconstructed images. (Images used with permission from [9].)

representation $\boldsymbol{h}^*$ and then obtain the reconstructed image $\hat{\boldsymbol{x}}$ via $\hat{\boldsymbol{x}} = \operatorname{argmax}_{\hat{x}} p(\hat{\boldsymbol{x}} \mid \boldsymbol{h}^*)$.

Figure 6 presents some examples. For the Multi-PIE data set [9], we trained the RBN with 200 latent variables; all face images are cropped and resized to 32 × 32 pixels. Similarly, we trained different RBNs on MNIST [17], UnityEye [4], and CAS-PEAL [3] data sets, respectively. When training the RBN on the MNIST data set, the number of hidden nodes is 50 for one hidden layer with the learning rate set to 0.01. For both the UnityEye and CAS-PEAL data sets, total hidden nodes are 200, image size is 60 × 36 and 30 × 30 accordingly. From the images in Figure 6, we can see that the DRBN is able to effectively capture data distribution and can produce good image reconstruction.

The last task for assessing the DRBN's data representation capability is through image generation. For this task, we use a pretrained DRBN model in the image reconstruction task.

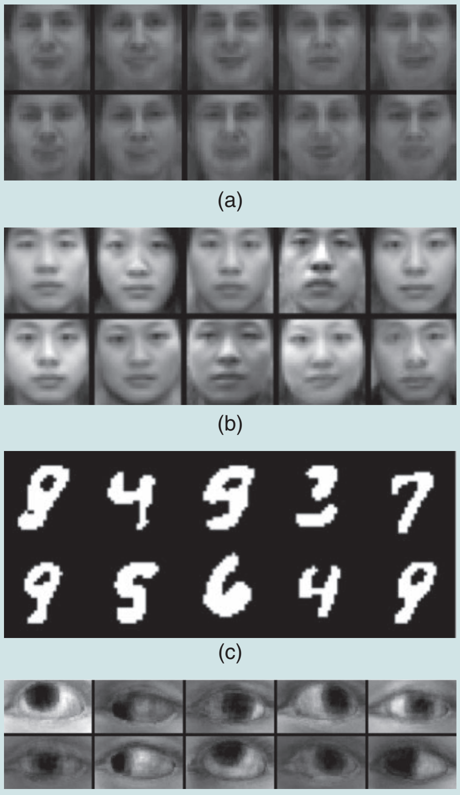

FIGURE 6. Examples of the image reconstruction for (a) Multi-PIE, (b) CAS-PEAL, (c) MNIST, (and d) UnityEye.

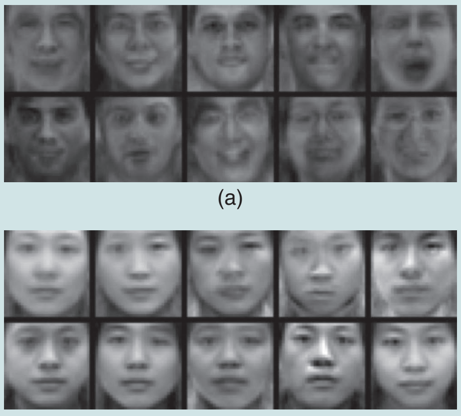

FIGURE 7. An example of the image generation experiment. Generated images from (a) Multi-PIE and (b) CAS-PEAL.

To generate realistic and meaningful images, we sample the latent variables from its prior probabilities. Given the values of the hidden nodes, images can be generated from sampling the conditional probability of visible nodes given latent variables. Examples of generated images from the Multi-PIE and CAS-PEAL data sets are shown in Figure 7.

Table 4. MAE of head pose angles in the BU data set.

Method	Yaw	Pitch	Roll
DMF	5.2	4.5	2.6
3D-Deform	4.3	6.2	3.2
MHPE	5.0	3.7	2.9
DVF+CNN	4.3	3.7	2.6
DRBN (IN) [21]	4.8	3.8	3.7
DRBN (CA)	5.4	5.8	3.5
DRBN (AugCA)	4.6	3.5	3.3

Table 5. Regression errors for eye-gaze estimation in the MPII data set.

Method	Yaw	Pitch	MAE	Std.
SVR [32]	–	–	6.6	0.6
ALR [18]	–	–	7.9	1.0
kNN [33]	–	–	7.2	0.8
CNN [39]	–	–	6.3	1.0
DRBN (IN)	5.8	3.7	7.6	1.1
DRBN (CA)	5.4	3.8	7.8	1.3
DRBN (AugCA)	4.9	3.6	7.1	1.2

Feature learning for regression

In this section, we applied the discriminative supervised learning for feature learning using the DRBN. Given the learned DRBN model, features in the top layer can be extracted via MAP inference. The extracted features will then be used for regression tasks. We study head-pose estimation (HPE) and eye-gaze estimation task with regression, respectively.

The HPE task is to predict three head-pose angles (roll, pitch, and yaw) through a regression function. The Boston University head-pose data set is used; the cropped faces are resized to 15×15 pixels. Two latent layers are used, with 50 latent variables each. The MAP configuration of the top layer is obtained through the IN, CA, and AugCA methods and is used as the features for a linear regression model. We compare the DRBN to three state-of-art model-based algorithms 1) three-dimensional (3-D) deformable HPE (3D-Deform) [37], 2) deformable model fitting (DMF) [30], and 3) monocular HPE (MHPE) [22] and one learning-based algorithm, distance vector field with CNNs (DVF+CNNs) [1], with the mean absolute errors (MAEs) reported in Table 4. The DRBN achieves comparable performance comparing to these competing algorithms. Note that the resolution of the face image used in the DRBN is much smaller (15×15 pixels) than other models (typically 320×240 pixels), and no 3-D information is used. This indicates the potential of using the DRBN for HPE in extremely low-resolution images.

For the task of eye-gaze estimation, we choose the MPII data set [39] because it covers a wide range of recording locations and times, illuminations, and eye appearances. The data set contains 213,659 images from 15 subjects. The goal of our experiment is to map the eye images to the pitch and yaw eye-gaze angles. The size of the cropped eye image is 36×60, yielding a 2,160-dimension vector. The DRBN has a 2160-200-200 structure. To evaluate different models, we use the "within-data set leave-one-subject-out evaluation," in which the eye images from one subject are used for testing and all the other images are used for training. The regression accuracies of different inference algorithms for each angle are given in Table 5. As competing algorithms, we include the CNN [39], k-nearest neighbor (kNN) [33], adaptive linear regression (ALR) [18], and support vector regression (SVR) [32]. The DRBN achieves comparable performance in terms of the mean average errors, demonstrating its effectiveness to capture the inherent patterns in the eye images. The experiment results show that DRBNs are able to learn high-quality features for regression tasks. Experimental results in the sections "Data Representation" and "Feature Learning for Regression" show that the DRBN can perform both generative and discriminative tasks, even though its discriminative performance may not match state-of-the-art CNN models.

Summary

In this article, we review different structures of deep directed generative models and their learning and inference algorithms. We focus on a specific version of deep generative models—the DRBN. Compared to other deep-learning models, the DRBN can better capture the data distribution because of its ability to explicitly capture the dependencies among the latent variables Various algorithms are reviewed and compared, including the efficient inference and learning methods of replacing the max-sum operation with the max-max operation and the augmented CA method for MAP inference. Extensive experiments on benchmark data sets for both data generation and classification tasks are presented, including image restoration, face reconstruction, HPE, and eye-gaze estimation. These experiments demonstrate the competitive performance of DRBNs for both data representation and feature-learning tasks.

Authors

Siqi Nie (niesiqi@gmail.com) received his bachelor's degree in electronic engineering from Tsinghua University, Beijing, China, in 2011 and his Ph.D. degree in electrical, computer, and systems engineering from Rensselaer Polytechnic Institute, Troy, New York, in December 2016. His research focuses on machine learning and its applications in computer vision. Specifically, he is interested in efficient learning and inference algorithms in probabilistic graphical models, including deep Bayesian networks and restricted Boltzmann machines. He has published more than ten conference and journal papers in the Conference and Workshop on Neural Information Processing Systems, the Association for the Advancement of Artificial Intelligence, *Computer Vision and Image Understanding*, and *International Journal of Approximate Reasoning*. Currently, he is a research scientist at Facebook.

Meng Zheng (zhengm3@rpi.edu) received her B.S. and M.S. degrees from the Department of Information and

Electronics, Beijing Institute of Technology, China, in 2013 and 2016, respectively. She is currently a Ph.D. student in the Department of Electrical, Computer, and Systems Engineering at Rensselaer Polytechnic Institute, Troy, New York.

Qiang Ji (qji@ecse.rpi.edu) received his M.S. degree from the University of Arizona, Tucson, in 1993, and his Ph.D. degree from the University of Washington, Seattle, in 1998, both in electrical engineering. He is currently a professor with the Department of Electrical, Computer, and Systems Engineering at Rensselaer Polytechnic Institute (RPI). From 2009 to 2010, he served as a program director at the National Science Foundation (NSF), Arlington, Virginia, where he managed NSF's computer vision and machine-learning programs. Currently, he serves as the director of the Intelligent Systems Laboratory at RPI. His research interests are in computer vision, probabilistic graphical models, machine learning, and their applications in various fields. He has published more than 200 papers in peer-reviewed journals and conferences and has received multiple awards for his work. He is an editor of several related IEEE and international journals and he has served as a general chair, program chair, technical area chair, and program committee member for numerous international conferences/workshops. He is a Fellow of the IEEE and the International Association for Pattern Recognition.

References

[1] S. Asteriadis, K. Karpouzis, and S. Kollias, "Visual focus of attention in non-calibrated environments using gaze estimation," *Int. J. Comput. Vis.*, vol. 107, no. 3, pp. 293–316, May 2014.

[2] B. Philip, "An architecture for deep, hierarchical generative models," in *Proc. Neural Information Processing Systems Conf.*, 2016, pp. 4826–4834.

[3] B. Liu, D. Zhou, X. Zhang, and W. Gao, "Introduction of the JDL large-scale facial database," in *Proc. 4th Chinese Conf. Biometrics Recognition*, 2003, pp. 118–121.

[4] L. Morency, P. Robinson, E. Wood, T. Baltrušaitis, and A. Bulling, Eds., "Learning an appearance-based gaze estimator from one million synthesised images," in *Proc. ACM Symp. Eye Tracking Research Applications*, 2016, pp. 131–138.

[5] M. Elad and M. Aharon, "Image denoising via sparse and redundant representations over learned dictionaries," *IEEE Trans. Image Process.*, vol. 15, no. 12, pp. 3736–3745, Dec. 2006.

[6] Z. Gan, C. Chen, R. Henao, D. Carlson, and L. Carin. "Scalable deep Poisson factor analysis for topic modeling," in *Proc. Int. Conf. Machine Learning*, Lille, France, 2015, pp. 1823–1832.

[7]. D. Geman and C. Yang, "Nonlinear image recovery with half-quadratic regularization," *IEEE Trans. Image Process.*, vol. 4, no. 7, pp. 932–946, July 1995.

[8] G. Ian, P.-A. Jean, M. Mehdi, X. Bing, W.-F. David, O. Sherjil, C. Aaron, and B. Yoshua, "Generative adversarial nets," in *Proc. Neural Information Processing Systems Conf.*, Montreal, Canada, 2014, pp. 2672–2680.

[9] R. Gross, I. Matthews, J. Cohn, T. Kanade, and S. Baker, "Multi-Pie," *Image Vis. Comput.*, vol. 28, no. 5, pp. 807–813, May 2010.

[10] H. Tian, L. Yang, Z. Song-Chun, and W. Ying Nian, "Alternating back-propagation for generator network," in *Proc. Assoc. for the Advancement of Artificial Intelligence,* 2017, pp. 1976–1984.

[11] G. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural Comput.*, vol. 18, no. 7, pp. 1527–1554, July 2006.

[12] G. Hinton and R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, July 2006.

[13] G. E. Hinton, P. Dayan, B. J. Frey, and R. M. Neal, "The "wake-sleep" algorithm for unsupervised neural networks," *Science*, vol. 268, no. 5214, pp. 1158–1161, May 1995.

[14] M. D. Hoffman, "Learning deep latent Gaussian models with Markov chain Monte Carlo," in *Proc. Int. Conf. Machine Learning*, 2017, pp. 1510–1519.

[15] D. P. Kingma and M. Welling, "Auto-encoding variational Bayes," in *Proc. Int. Conf. Learning Representations*, Banff, Canada, 2014.

[16] A. B. L. Larsen, S. K Sønderby, H. Larochelle, and O. Winther, "Autoencoding beyond pixels using a learned similarity metric," in *Proc. 33rd Int. Conf. Machine Learning,* 2016, pp. 1558–1566.

[17] L. Bottou, Y. Bengio, Y. LeCun, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[18] L. Feng, S. Yusuke, O. Takahiro, and S. Yoichi, "Adaptive linear regression for appearance-based gaze estimation," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 36, no. 10, pp. 2033–2046, Oct. 2014.

[19] D. Martin, C. Fowlkes, D. Tal, and J. Malik, "A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics," in *Proc. 8th IEEE Int. Conf. Computer Vision*, vol. 2, Vancouver, Canada, 2001, pp. 416–423.

[20] M. Lars, N. Sebastian, and G. Andreas, "Adversarial variational bayes: Unifying variational autoencoders and generative adversarial networks," arXiv Preprint arXiv:1701.04722, 2017.

[21] A. Mnih and K. Gregor, "Neural variational inference and learning in belief networks," in *Proc. 31st Int. Conf. Machine Learning*, Beijing, China, 2014, pp. 1791–1799.

[22] L. Morency, J. Whitehill, and J. Movellan, "Monocular head pose estimation using generalized adaptive view-based appearance model," *Image Vis. Comput.*, vol. 28, no. 5, pp. 754–761, May 2010.

[23] I. Murray and R. R. Salakhutdinov, "Evaluating probabilities under high-dimensional latent variable models," in *Proc. Neural Information Processing Systems Conf.*, Vancouver, Canada, 2009, pp. 1137–1144.

[24] R. M. Neal, "Connectionist learning of belief networks," *Artif. Intell.*, vol. 56, no. 1, pp. 71–113, July 1992.

[25] R. Rajesh, G. Sean, and B. David, "Black box variational inference," in *Proc. Artificial Intelligence Statistics Conf.*, 2014, pp. 814–822.

[26] D. J. Rezende, S. Mohamed, and D. Wierstra. "Stochastic backpropagation and approximate inference in deep generative models," in *Proc. 31st Int. Conf. Machine Learning*, Beijing, China, 2014, pp. 1278–1286.

[27] F. Rijmen, "Bayesian networks with a logistic regression model for the conditional probabilities," *Int. J. Approx. Reason*, vol. 48, no. 2, pp. 659–666, June 2008.

[28] S. Roth and M. J. Black. "Fields of experts: A framework for learning image priors," in *Proc. IEEE Conf. Computer Vision Pattern Recognition*, vol. 2, 2005, pp. 860–867.

[29] R. Salakhutdinov and G. E. Hinton. "Deep Boltzmann machines," in *Proc. Conf. Artificial Intelligence Statistics*, Clearwater Beach, FL, 2009, pp. 448–455.

[30] J. Saragih, S. Lucey, and J. Cohn, "Deformable model fitting by regularized landmark mean-shift," *Int. J. Comput. Vis.*, vol. 91, no. 2, pp. 200–215, Jan. 2011.

[31] L. K. Saul, T. Jaakkola, and M. I. Jordan, "Mean field theory for sigmoid belief networks," *J. Artif. Intell. Res.*, vol. 4, no. 61, pp. 76, Jan. 1996.

[32] T. Schneider, B. Schauerte, and R. Stiefelhagen,"Manifold alignment for person independent appearance-based gaze estimation," in *Proc. 22nd Int. Conf. Pattern Recognition*, Stockholm, Sweden, 2014, pp. 1167–1172.

[33] S. Yusuke, M. Yasuyuki, and S. Yoichi, "Learning-by-synthesis for appearance-based 3D gaze estimation," in *Proc. IEEE Conf. Computer Vision Pattern Recognition*, Columbus, OH, 2014, pp. 1821–1828.

[34] Y. Tang, G. E. Hinton, and R. Salakhutdinov, "Deep mixtures of factor analysers," in *Proc. 29th Int. Conf. Machine Learning*, Edinburgh, Scotland, 2012, pp. 505–512.

[35] T. Yichuan, S. Ruslan, and H. Geoffrey, "Robust Boltzmann machines for recognition and denoising," in *Proc. 25th IEEE Conf. Computer Vision Pattern Recognition*, Providence, RI, 2012, pp. 2264–2271.

[36] A. van den Oord and B. Schrauwen, "Factoring variations in natural images with deep Gaussian mixture models," in *Proc. Neural Information Processing Systems Conf.*, Montreal, Canada, 2014, pp. 3518–3526.

[37] F. Vicente, Z. Huang, X. Xiong, F. De la Torre, W. Zhang, and D. Levi, "Driver gaze tracking and eyes off the road detection system," *IEEE Trans. Intell. Transp. Syst.*, vol. 16, no. 4, pp. 2014–2027, Aug. 2015.

[38] W. John, G. Arvind, R. Shankar, P. Yigang, and M. Yi, "Robust principal component analysis: Exact recovery of corrupted low-rank matrices via convex optimization," in *Proc. Neural Information Processing Systems Conf.*, Vancouver, Canada, 2009, pp. 2080–2088.

[39] Z. Xucong, S. Yusuke, F. Mario, and B. Andreas, "Appearance-based gaze estimation in the wild," in *Proc. IEEE Conf. Computer Vision Pattern Recognition*, Boston, MA, 2015, pp. 4511–4520.

[40] D. Zoran and Y. Weiss, "From learning models of natural image patches to whole image restoration," in *Proc. IEEE Int. Conf. Computer Vision*, Barcelona, Spain, 2011, pp. 479–486.