Towards Bayesian Deep Learning: A Framework and Some Existing Methods

Hao Wang and Dit-Yan Yeung, *Senior Member, IEEE*

Abstract—While perception tasks such as visual object recognition and text understanding play an important role in human intelligence, subsequent tasks that involve inference, reasoning, and planning require an even higher level of intelligence. The past few years have seen major advances in many perception tasks using deep learning models. For higher-level inference, however, probabilistic graphical models with their Bayesian nature are still more powerful and flexible. To achieve integrated intelligence that involves both perception and inference, it is naturally desirable to tightly integrate deep learning and Bayesian models within a principled probabilistic framework, which we call *Bayesian deep learning*. In this unified framework, the perception of text or images using deep learning can boost the performance of higher-level inference and in return, the feedback from the inference process is able to enhance the perception of text or images. This paper proposes a general framework for *Bayesian deep learning* and reviews its recent applications on recommender systems, topic models, and control. In this paper, we also discuss the relationship and differences between Bayesian deep learning and other related topics such as the Bayesian treatment of neural networks.

Index Terms—Artificial intelligence, data mining, Bayesian networks, neural networks, deep learning, machine learning

1 Introduction

Deep learning has achieved significant success in many perception tasks including *seeing* (visual object recognition), *reading* (text understanding), and *hearing* (speech recognition). These are undoubtedly fundamental tasks for a functioning comprehensive artificial intelligence (AI) or data engineering (DE) system. However, in order to build a real AI/DE system, simply being able to see, read, and hear is far from enough. It should, above all, possess the ability to *think*.

Take medical diagnosis as an example. Besides *seeing* visible symptoms (or medical images from CT) and *hearing* descriptions from patients, a doctor has to look for relations among all the symptoms and preferably infer the corresponding etiology. Only after that can the doctor provide medical advice for the patients. In this example, although the abilities of *seeing* and *hearing* allow the doctor to acquire information from the patients, it is the *thinking* part that defines a doctor. Specifically, the ability to *think* here could involve causal inference, logic deduction, and dealing with uncertainty, which is apparently beyond the capability of conventional deep learning methods. Fortunately, another type of models, probabilistic graphical models (PGM), excels at causal inference and dealing with uncertainty. The problem is that PGM is not as good as deep learning models at perception tasks. To address the problem, it is, therefore, a natural choice to tightly integrate deep learning and PGM within a principled probabilistic framework, which we call *Bayesian deep learning* (BDL) in this paper.

With the tight and principled integration in BDL, perception tasks and inference tasks are regarded as a whole and can benefit from each other. In the example above, being able to see the medical image could help with the doctor's diagnosis and inference. On the other hand, diagnosis and inference can in return help with understanding the medical image. Suppose a doctor is not sure what a dark spot in a medical image is. However, if she is able to *infer* the etiology of the symptoms and disease, it can help her better decide whether the dark spot is a tumor or not.

As another example, to achieve high accuracy in recommender systems (RS) [1], [39], [40], [50], [67], we need to fully understand the content of the items (e.g., documents and movies) [46], analyze the profile and preferences of users [70], [73], and evaluate the similarity among the users [3], [11], [29]. Deep learning is good at the first subtask while PGM excels at the other two. Besides the fact that better understanding of item content would help with the analysis of user profiles, the estimated similarity among users could also provide valuable information for understanding item content in return. In order to fully utilize this bidirectional effect to boost recommendation accuracy, we might wish to unify deep learning and PGM in one single principled probabilistic framework, as seen in [67].

Besides recommender systems, the need for BDL may also arise when we are dealing with the control of non-linear dynamic systems with raw images as input. Consider controlling a complex dynamical system according to the live video stream received from a camera. This problem can be transformed into iteratively performing two tasks, *perception* from raw images and *control* based on dynamic models. The perception task can be taken care of using multiple layers of simple nonlinear transformation (deep learning) while the control task usually needs more sophisticated models like hidden Markov models and Kalman filters [22]. The feedback loop is then completed by the fact that actions chosen by the

• *The authors are with the Department of Computer Science and Engineering, The Hong Kong University of Science and Technology, Kowloon, Hong Kong. E-mail: {hwangaz, dyyeung}@cse.ust.hk.*

Manuscript received 26 May 2016; revised 15 July 2016; accepted 21 Aug. 2016. Date of publication 7 Sept. 2016; date of current version 2 Nov. 2016. Recommended for acceptance by S. Yan.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TKDE.2016.2606428

Authorized licensed use limited to: Rutgers University Libraries. Downloaded on September 15,2024 at 02:31:03 UTC from IEEE Xplore. Restrictions apply.
1041-4347 © 2016 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

control model can affect the received video stream in return. To enable an effective iterative process between the perception task and the control task, we need two-way information exchange between them. The perception component would be the basis on which the control component estimates its states and the control component with a built-in dynamic model would be able to predict the future trajectory (images). In such cases, BDL is a suitable choice [69].

As mentioned in the examples above, BDL is particularly useful for tasks that involve both understanding of content (e.g., text, images, and videos) and inference/reasoning among variables. In such complex tasks, the perception component of BDL is responsible for the understanding of the content, and the task-specific component (e.g., the control component in dynamical systems) models the probabilistic relationship among different variables. Furthermore, the interaction between these two components creates synergy and further boosts the performance.

Apart from the major advantage of BDL providing a principled way of unifying deep learning and PGM, another benefit comes from the implicit regularization built into BDL. Through imposing a prior on hidden units, parameters defining a neural network, or the model parameters specifying the causal inference, to some degree BDL can avoid overfitting, especially when there is not sufficient data. Usually, a BDL model consists of two components: (1) a *perception component* that is a Bayesian formulation of a certain type of neural networks (NN) and (2) a *task-specific component* that describes the relationship among different hidden or observed variables using PGM. Regularization is crucial for them both. Neural networks usually have large numbers of free parameters that need to be regularized properly. Regularization techniques such as weight decay and dropout [57] are shown to be effective in improving performance of neural networks and they both have Bayesian interpretations [15]. In terms of the task-specific component, expert knowledge or prior information, as a kind of regularization, can be incorporated into the model through the prior we imposed to guide the model when data are scarce.

Yet another advantage of using BDL for complex tasks (tasks that need both perception and inference) is that it provides a principled Bayesian approach of handling parameter uncertainty. When BDL is applied to complex tasks, there are *three kinds of parameter uncertainties* that need to be taken into account:

1) Uncertainty about the neural network parameters.
2) Uncertainty about the task-specific parameters.
3) Uncertainty about the exchange of information between the perception component and the task-specific component.

Through representing the unknown parameters using distributions instead of point estimates, BDL offers a promising framework to handle these three kinds of uncertainty in a unified way. It is worth noting that the third uncertainty could only be handled under a unified framework such as BDL. If we train the perception component and the task-specific component separately, it is equivalent to assuming *no uncertainty* when *exchanging information* between the two components.

Of course, there are challenges when applying BDL to real-world tasks. (1) First, it is nontrivial to design an efficient Bayesian formulation of neural networks with reasonable time complexity. This line of work has been pioneered by [25], [41], [44], but it has not been widely adopted due to its lack of scalability. Fortunately, some recent advances in this direction [2], [9], [23], [34], [66] seem to shed light on the practical adoption of Bayesian neural networks (BNN).[1] (2) The second challenge is to ensure efficient and effective information exchange between the perception component and the task-specific component. Ideally both the first-order and second-order information (e.g., the mean and the variance) should be able to flow back and forth between the two components. A natural way is to represent the perception component as a PGM and seamlessly connect it to the task-specific PGM, as done in [17], [64], [67].

In this paper, we aim to give a comprehensive overview of BDL models for applications like recommender systems, topic models (and representation learning), and control. The rest of the paper is organized as follows: In Section 2, we provide a review of some basic deep learning models. Section 3 covers the main concepts and techniques for PGM. These two sections serve as the background for BDL, and the next section, Section 4, proposes a unified BDL framework and surveys the BDL models applied to areas such as recommender systems and topic models. Section 5 discusses some future research issues and concludes the paper.

2 Deep Learning

Deep learning normally refers to neural networks with more than two layers. To better understand deep learning, here we start with the simplest type of neural networks, multilayer perceptrons (MLP), as an example to show how conventional deep learning works. After that, we will review several other types of deep learning models based on MLP.

2.1 Multilayer Perceptron

Essentially a multilayer perceptron is a sequence of parametric nonlinear transformations. Suppose we want to train a multilayer perceptron to perform a regression task which maps a vector of M dimensions to a vector of D dimensions. We denote the input as a matrix $\mathbf{X}_0$ (0 means it is the 0th layer of the perceptron). The jth row of $\mathbf{X}_0$, denoted as $\mathbf{X}_{0,j*}$, is an M-dimensional vector representing one data point. The target (the output we want to fit) is denoted as $\mathbf{Y}$. Similarly $\mathbf{Y}_{j*}$ denotes a D-dimensional row vector. The problem of learning an L-layer multilayer perceptron can be formulated as the following optimization problem:

$$\begin{aligned} \min_{\{\mathbf{W}_l\},\{\mathbf{b}_l\}} \; & \|\mathbf{X}_L - \mathbf{Y}\|_F + \lambda \sum_l \|\mathbf{W}_l\|_F^2 \\ \text{subject to} \; & \mathbf{X}_l = \sigma(\mathbf{X}_{l-1}\mathbf{W}_l + \mathbf{b}_l), l = 1, \ldots, L-1 \\ & \mathbf{X}_L = \mathbf{X}_{L-1}\mathbf{W}_L + \mathbf{b}_L, \end{aligned}$$

where $\sigma(\cdot)$ is an element-wise sigmoid function for a matrix and $\sigma(x) = \frac{1}{1+\exp(-x)}$. λ is a regularization parameter and $\|\cdot\|_F$ denotes the Frobenius norm. The purpose of imposing $\sigma(\cdot)$ is to allow nonlinear transformation. Normally other

1. Here, we refer to Bayesian treatment of neural networks as Bayesian neural networks. The other term, Bayesian deep learning, is retained to refer to complex Bayesian models with both a perception component and a task-specific component.

Authorized licensed use limited to: Rutgers University Libraries. Downloaded on September 15,2024 at 02:31:03 UTC from IEEE Xplore. Restrictions apply.

transformations like $\tanh(x)$ and $\max(0, x)$ can be used as alternatives of the sigmoid function.

Here $\mathbf{X}_l$ $(l = 1, 2, \ldots, L-1)$ is the hidden units. As we can see, $\mathbf{X}_L$ can be easily computed once $\mathbf{X}_0$, $\mathbf{W}_l$, and $\mathbf{b}_l$ are given. Since $\mathbf{X}_0$ is given by the data, we only need to learn $\mathbf{W}_l$ and $\mathbf{b}_l$ here. Usually this is done using backpropagation and stochastic gradient descent (SGD). The key is to compute the gradients of the objective function with respect to $\mathbf{W}_l$ and $\mathbf{b}_l$. If we denote the value of the objective function as E, we can compute the gradients using the chain rule as

$$\frac{\partial E}{\partial \mathbf{X}_L} = 2(\mathbf{X}_L - \mathbf{Y}) \tag{1}$$

$$\frac{\partial E}{\partial \mathbf{X}_l} = \left(\frac{\partial E}{\partial \mathbf{X}_{l+1}} \circ \mathbf{X}_{l+1} \circ (1 - \mathbf{X}_{l+1})\right)\mathbf{W}_{l+1} \tag{2}$$

$$\frac{\partial E}{\partial \mathbf{W}_l} = \mathbf{X}_{l-1}^T\left(\frac{\partial E}{\partial \mathbf{X}_l} \circ \mathbf{X}_l \circ (1 - \mathbf{X}_l)\right) \tag{3}$$

$$\frac{\partial E}{\partial \mathbf{b}_l} = mean\left(\frac{\partial E}{\partial \mathbf{X}_l} \circ \mathbf{X}_l \circ (1 - \mathbf{X}_l), 1\right), \tag{4}$$

where $l = 1, \ldots, L$ and the regularization terms are omitted. The element-wise product is denoted as $\circ$ and $mean(\cdot, 1)$ is the matlab operation on matrices. In practice, we only use a small part of the data (e.g., 128 data points) to compute the gradients for each update. This is called stochastic gradient descent.

As we can see, in conventional deep learning models, only $\mathbf{W}_l$ and $\mathbf{b}_l$ are free parameters, which we will update in each iteration of the optimization. $\mathbf{X}_l$ is not a free parameter since it can be computed exactly if $\mathbf{W}_l$ and $\mathbf{b}_l$ are given.

2.2 Autoencoders

An autoencoder (AE) is a feedforward neural network to encode the input into a more compact representation and reconstruct the input with the learned representation. In its simplest form, an autoencoder is no more than a multilayer perceptron with a bottleneck layer (a layer with a small number of hidden units) in the middle. The idea of autoencoders has been around for decades [10], [20], [35] and abundant variants of autoencoders have been proposed to enhance representation learning including sparse AE [48], contractive AE [51], and denoising AE [59]. For more details, please refer to a nice recent book on deep learning [20]. Here we introduce a kind of multilayer denoising AE, known as stacked denoising autoencoders (SDAE), both as an example of AE variants and as background for its applications on BDL-based recommender systems in Section 4.

SDAE [59] is a feedforward neural network for learning representations (encoding) of the input data by learning to predict the clean input itself in the output, as shown in Fig. 1. The hidden layer in the middle, i.e., $\mathbf{X}_2$ in the figure, can be constrained to be a bottleneck to learn compact representations. The difference between traditional AE and SDAE is that the input layer $\mathbf{X}_0$ is a *corrupted* version of the *clean* input data. Essentially an SDAE solves the following optimization problem:

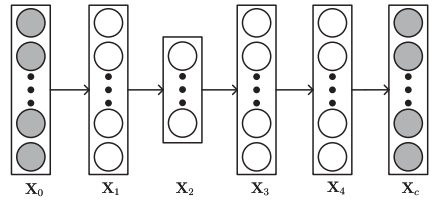


Fig. 1. A 2-layer SDAE with $L = 4$.

$$\min_{\{\mathbf{W}_l\},\{\mathbf{b}_l\}} \|\mathbf{X}_c - \mathbf{X}_L\|_F^2 + \lambda \sum_l \|\mathbf{W}_l\|_F^2$$
$$\text{subject to } \mathbf{X}_l = \sigma(\mathbf{X}_{l-1}\mathbf{W}_l + \mathbf{b}_l), l = 1, \ldots, L-1$$
$$\mathbf{X}_L = \mathbf{X}_{L-1}\mathbf{W}_L + \mathbf{b}_L,$$

Here SDAE can be regarded as a multilayer perceptron for regression tasks described in the previous section. The input $\mathbf{X}_0$ of the MLP is the corrupted version of the data and the target $\mathbf{Y}$ is the clean version of the data $\mathbf{X}_c$. For example, $\mathbf{X}_c$ can be the raw data matrix, and we can randomly set 30 percent of the entries in $\mathbf{X}_c$ to 0 and get $\mathbf{X}_0$. In a nutshell, SDAE learns a neural network that takes the noisy data as input and recovers the clean data in the last layer. This is what 'denoising' means. Normally, the output of the middle layer, i.e., $\mathbf{X}_2$ in Fig. 1, would be used to compactly represent the data.

2.3 Other Deep Learning Models

Other commonly used deep learning models include convolutional neural networks (CNN) [31], [36], which apply convolution operators and pooling operators to process image or video data, and recurrent neural networks (RNN) [20], [26], which use recurrent computation to imitate human memory, and restricted Boltzmann machines (RBM) [24], which are undirected probabilistic neural networks with binary hidden and visible layers. Note that there is a vast literature on deep learning and neural networks. The introduction in this section intends to serve only as the background of BDL. Readers are referred to [20] for a comprehensive survey and more details.

3 Probabilistic Graphical Models

Probabilistic Graphical Models use diagrammatic representations to describe random variables and relationships among them. Similar to a graph that contains nodes (vertices) and links (edges), PGM has nodes to represent random variables and links to express probabilistic relationships among them.

3.1 Models

As pointed out in [5], there are two main types of PGMs, directed PGMs (also known as Bayesian networks) and undirected PGMs (also known as Markov random fields), although there exist hybrid ones. In this paper we mainly focus on directed PGMs.[2] For details on undirected PGMs, readers are referred to [5].

A classic example of a PGM would be latent Dirichlet allocation (LDA), which is used as a topic model to analyze the generation of words and topics in documents. Usually PGM comes with a graphical representation of the model and a generative process to depict the story of how the

2. For convenience, PGM stands for directed PGM in this paper unless specified otherwise.

Authorized licensed use limited to: Rutgers University Libraries. Downloaded on September 15,2024 at 02:31:03 UTC from IEEE Xplore. Restrictions apply.

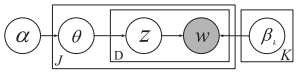


Fig. 2. The probabilistic graphical model for LDA, J is the number of documents and D is the number of words in a document.

random variables are generated step by step. Fig. 2 shows the graphical model for LDA and the corresponding generative process is as follows:

- For each document j ($j = 1, 2, \ldots, J$),
  1) Draw topic proportions $\theta_j \sim \text{Dirichlet}(\alpha)$.
  2) For each word w_{jn} of item (paper) $\mathbf{w}_j$,
     a) Draw topic assignment $z_{jn} \sim \text{Mult}(\theta_j)$.
     b) Draw word $w_{jn} \sim \text{Mult}(\beta_{z_{jn}})$.

The generative process above gives the story of how the random variables are generated. In the graphical model in Fig. 2, the shaded node denotes observed variables while the others are latent variables (θ and $\mathbf{z}$) or parameters (α and β). As we can see, once the model is defined, learning algorithms can be applied to automatically learn the latent variables and parameters.

Due to its Bayesian nature, PGM like LDA is easy to extend to incorporate other information or to perform other tasks. For example, after LDA, different variants of topic models based on it have been proposed. The authors in [7], [61] proposed to incorporate temporal information and [6] extends LDA by assuming correlations among topics. To make it possible to process large datasets, [27] extends LDA from the batch mode to the online setting. On recommender systems, [60] extends LDA to incorporate rating information and make recommendations. This model is then further extended to incorporate social information [49], [62], [63].

3.2 Inference and Learning

Strictly speaking, the process of finding the parameters (e.g., α and β in Fig. 2) is called learning and the process of finding the latent variables (e.g., θ and $\mathbf{z}$ in Fig. 2) given the parameters is called inference. However, given only the observed variables (e.g., $\mathbf{w}$ in Fig. 2), learning and inference are often intertwined. Usually, the learning and inference of LDA would alternate between the updates of latent variables (which correspond to inference) and the updates of the parameters (which correspond to learning). Once the learning and inference of LDA is completed, we would have the parameters α and β. If a new document arrives, we can now fix the learned α and β and then perform inference alone to find the topic proportions θ_j of the new document.[3]

As in LDA, various learning and inference algorithms are available for each PGM. Among them, the most cost-effective one is probably maximum a posteriori (MAP), which amounts to maximizing the posterior probability of the latent variable. Using MAP, the learning process is equivalent to minimizing (or maximizing) an objective function with regularization. One famous example is the probabilistic matrix factorization (PMF) [53]. The learning of the graphical model in PMF is equivalent to the factorization of a large matrix into two low-rank matrices with L2 regularization.

3. For convenience, we use 'learning' to represent both 'learning and inference' in the following text.

MAP, as efficient as it is, gives us only *point estimates* of latent variables (and parameters). In order to take the uncertainty into account and harness the full power of Bayesian models, one would have to resort to Bayesian treatments such as variational inference and Markov chain Monte Carlo (MCMC). For example, the original LDA uses variational inference to approximate the true posterior with factorized variational distributions [8]. Learning of the latent variables and parameters then boils down to minimizing the KL-divergence between the variational distributions and the true posterior distributions. Besides variational inference, another choice for a Bayesian treatment is to use MCMC. For example, MCMC algorithms such as [47] have been proposed to learn the posterior distributions of LDA.

4 Bayesian Deep Learning

With the background on deep learning and PGM, we are now ready to introduce the general framework and some concrete examples of BDL. Specifically, in this section we will list some recent BDL models with applications on recommender systems and topic models. A summary of these models is shown in Table 1.

4.1 General Framework

As mentioned in Section 1, BDL is a principled probabilistic framework with two seamlessly integrated components: a *perception component* and a *task-specific component*.

PGM for BDL. Fig. 3 shows the PGM of a simple BDL model as an example. The part inside the red rectangle on the left represents the perception component and the part inside the blue rectangle on the right is the task-specific component. Typically, the perception component would be a probabilistic formulation of a deep learning model with multiple nonlinear processing layers represented as a chain structure in the PGM. While the nodes and edges in the perception component are relatively simple, those in the task-specific component often describe more complex distributions and relationships among variables (as in LDA).

Three Sets of Variables. There are three sets of variables in a BDL model: perception variables, hinge variables, and task variables: (1) In this paper, we use Ω_p to denote the set of perception variables (e.g., $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{C}$ in Fig. 3), which are the variables in the perception component. Usually Ω_p would include the weights and neurons in the probabilistic formulation of a deep learning model. (2) We use Ω_h to denote the set of hinge variables (e.g., $\mathbf{J}$ in Fig. 3). These variables directly interact with the perception component from the task-specific component. Table 1 shows the set of hinge variables Ω_h for each listed BDL models. (3) The set of task variables (e.g., $\mathbf{G}$, $\mathbf{I}$, and $\mathbf{H}$ in Fig. 3), i.e., variables in the task-specific component without direct relation to the perception component, is denoted as Ω_t.

The I.I.D. Requirement. Note that hinge variables are always in the task-specific component. Normally, the connections between hinge variables Ω_h and the perception component (e.g., $\mathbf{C} \rightarrow \mathbf{J}$ in Fig. 3) should be i.i.d. for convenience of parallel computation in the perception component. For example, each row in $\mathbf{J}$ is related to only one corresponding row in $\mathbf{C}$. Although it is not mandatory in BDL models, meeting this requirement would significantly increase the efficiency of parallel computation in model training.

Authorized licensed use limited to: Rutgers University Libraries. Downloaded on September 15,2024 at 02:31:03 UTC from IEEE Xplore. Restrictions apply.

TABLE 1
Summary of BDL Models. Ω_h is the Set of Hinge Variables Mentioned in Section 4.1

Applications	Models	Ω_h	Variance of Ω_h	MAP	Gibbs Sampling	SG Thermostats
Recommender Systems	CDL	$\{\mathbf{V}\}$	Hyper-Variance	✓		
	Bayesian CDL	$\{\mathbf{V}\}$	Hyper-Variance		✓	
	Marginalized CDL	$\{\mathbf{V}\}$	Learnable Variance	✓		
	Symmetric CDL	$\{\mathbf{V}, \mathbf{U}\}$	Learnable Variance	✓		
	Collaborative Deep Ranking	$\{\mathbf{V}\}$	Hyper-Variance	✓		
Topic Models	Relational SDAE	$\{\mathbf{S}\}$	Hyper-Variance	✓		
	DPFA-SBN	$\{\mathbf{X}\}$	Zero-Variance		✓	✓
	DPFA-RBM	$\{\mathbf{X}\}$	Zero-Variance		✓	✓

$\mathbf{V}$ and $\mathbf{U}$ are the item latent matrix and the user latent matrix (Section 4.2.1). $\mathbf{S}$ is the relational latent matrix (Section 4.3.1), and $\mathbf{X}$ is the content matrix (Section 4.3.2).

Joint Distribution Decomposition. If the edges between the two components *point towards* Ω_h (as shown in Fig. 3, where $\Omega_p = \{\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}, \mathbf{E}, \mathbf{F}\}$, $\Omega_h = \{\mathbf{J}\}$, and $\Omega_t = \{\mathbf{I}, \mathbf{G}, \mathbf{H}\}$), the joint distribution of all variables can be written as

$$p(\Omega_p, \Omega_h, \Omega_t) = p(\Omega_p)p(\Omega_h|\Omega_p)p(\Omega_t|\Omega_h). \tag{5}$$

If the edges between the two components *originate from* Ω_h (similar to Fig. 3 except that the edge points from $\mathbf{J}$ to $\mathbf{C}$), the joint distribution of all variables can be written as

$$p(\Omega_p, \Omega_h, \Omega_t) = p(\Omega_t)p(\Omega_h|\Omega_t)p(\Omega_p|\Omega_h). \tag{6}$$

Apparently, it is possible for BDL to have some edges between the two components pointing towards Ω_h and some originating from Ω_h, in which case the decomposition of the joint distribution would be more complex.

Variance Related to Ω_h. As mentioned in Section 1, one of the motivations for BDL is to model the *uncertainty of exchanging information* between the perception component and the task-specific component, which boils down to modeling the uncertainty related to Ω_h. For example, this kind of uncertainty is reflected in the variance of the conditional density $p(\Omega_h|\Omega_p)$ in Equation (5).[4] According to the degree of flexibility, there are three types of variance for Ω_h (for simplicity we assume the joint likelihood of BDL is Equation (5), $\Omega_p = \{p\}$, $\Omega_h = \{h\}$, and $p(\Omega_h|\Omega_p) = \mathcal{N}(h|p, s)$ in our example):

- *Zero-Variance.* Zero-Variance (ZV) assumes no uncertainty during the information exchange between the two components. In the example, zero-variance means directly setting s to 0.
- *Hyper-Variance.* Hyper-Variance (HV) assumes that uncertainty during the information exchange is defined through hyperparameters. In the example, HV means that s is a hyperparameter that is manually tuned.
- *Learnable Variance.* Learnable Variance (LV) uses learnable parameters to represent uncertainty during the information exchange. In the example, s is the learnable parameter.

As shown above, we can see that in terms of model flexibility, LV > HV > ZV. Normally, if the models are properly regularized, an LV model would outperform an HV model, which is superior to a ZV model. In Table 1, we show the types of variance for Ω_h in different BDL models. Note that although each model in the table has a specific type, one can always adjust the models to devise their counterparts of other types. For example, while CDL in the table is an HV model, we can easily adjust $p(\Omega_h|\Omega_p)$ in CDL to devise its ZV and LV counterparts. In [67], authors compare the performance of an HV CDL and a ZV CDL and finds that the former performs significantly better, meaning that sophisticatedly modeling uncertainty between two components is essential for performance.

Learning Algorithms. Due to the nature of BDL, practical learning algorithms need to meet these criteria:

1) They should be online algorithms in order to scale well for large datasets.
2) They should be efficient enough to scale linearly with the number of free parameters in the perception component.

Criterion (1) implies that conventional variational inference or MCMC methods are not applicable. Usually an online version of them is needed [28]. Most SGD-based methods do not work either unless only MAP inference (as opposed to Bayesian treatments) is performed. Criterion (2) is needed because there are typically a large number of free parameters in the perception component. This means methods based on Laplace approximation [41] are not realistic since they involve the computation of a Hessian matrix that scales quadratically with the number of free parameters.

4.2 Bayesian Deep Learning for Recommender Systems

Despite the successful applications of deep learning on natural language processing and computer vision, very few attempts have been made to develop deep learning models for CF. The authors in [54] use restricted Boltzmann machines instead of the conventional matrix factorization formulation to perform CF and [19] extends this work by incorporating user-user and item-item correlations. Although these methods involve both deep learning and CF, they actually belong to CF-based methods because they

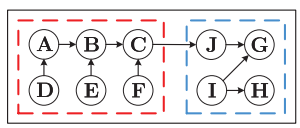


Fig. 3. The PGM for an example BDL. The red rectangle on the left indicates the perception component, and the blue rectangle on the right indicates the task-specific component. The hinge variable $\Omega_h = \{\mathbf{J}\}$.

4. For models with the joint likelihood decomposed as in Equation (6), the uncertainty is reflected in the variance of $p(\Omega_p|\Omega_h)$.

Authorized licensed use limited to: Rutgers University Libraries. Downloaded on September 15,2024 at 02:31:03 UTC from IEEE Xplore. Restrictions apply.

do not incorporate content information as in CTR [60], which is crucial for accurate recommendation. The authors in [52] use low-rank matrix factorization in the last weight layer of a deep network to significantly reduce the number of model parameters and speed up training, however it is for classification instead of recommendation tasks. On music recommendation, [45], [68] directly use conventional CNN or deep belief networks (DBN) to assist representation learning for content information, but the deep learning components of their models are deterministic without modeling the noise and hence they are less robust. The models achieve performance boost mainly by loosely coupled methods without exploiting the interaction between content information and ratings. Besides, the CNN is linked directly to the rating matrix, which means the models will perform poorly due to serious overfitting when the ratings are sparse.

4.2.1 Collaborative Deep Learning

To address the challenges above, a hierarchical Bayesian model called collaborative deep learning (CDL) as a novel tightly coupled method for RS is introduced in [67]. Based on a Bayesian formulation of SDAE, CDL tightly couples deep representation learning for the content information and collaborative filtering for the rating (feedback) matrix, allowing two-way interaction between the two. Experiments show that CDL significantly outperforms the state of the art.

In the following text, we will start with the introduction of the notation used during our presentation of CDL. After that we will review the design and learning of CDL.

Notation and Problem Formulation. Similar to the work in [60], the recommendation task considered in CDL takes implicit feedback [30] as the training and test data. The entire collection of J items (articles or movies) is represented by a J-by-B matrix $\mathbf{X}_c$, where row j is the bag-of-words vector $\mathbf{X}_{c,j*}$ for item j based on a vocabulary of size B. With I users, we define an I-by-J binary rating matrix $\mathbf{R} = [\mathbf{R}_{ij}]_{I\times J}$. For example, in the dataset *citeulike-a* [60], [62], [67] $\mathbf{R}_{ij} = 1$ if user i has article j in his or her personal library and $\mathbf{R}_{ij} = 0$ otherwise. Given part of the ratings in $\mathbf{R}$ and the content information $\mathbf{X}_c$, the problem is to predict the other ratings in $\mathbf{R}$. Note that although CDL in its current form focuses on movie recommendation (where plots of movies are considered as content information) and article recommendation like [60] in this section, it is general enough to handle other recommendation tasks (e.g., tag recommendation).

Matrix $\mathbf{X}_c$ plays the role of clean input to the SDAE while the noise-corrupted matrix, also a J-by-B matrix, is denoted by $\mathbf{X}_0$. The output of layer l of the SDAE is denoted by $\mathbf{X}_l$ which is a J-by-K_l matrix, where K_l is the number of units in layer l. Similar to $\mathbf{X}_c$, row j of $\mathbf{X}_l$ is denoted by $\mathbf{X}_{l,j*}$. $\mathbf{W}_l$ and $\mathbf{b}_l$ are the weight matrix and bias vector, respectively, of layer l, $\mathbf{W}_{l,*n}$ denotes column n of $\mathbf{W}_l$, and L is the number of layers. For convenience, we use $\mathbf{W}^+$ to denote the collection of all layers of weight matrices and biases. Note that an $L/2$-layer SDAE corresponds to an L-layer network.

Generalized Bayesian SDAE. Following the introduction of SDAE in Section 2.2, if we assume that both the clean input $\mathbf{X}_c$ and the corrupted input $\mathbf{X}_0$ are observed, similar to [4], [5], [12], [41], we can define the following generative process of generalized Bayesian SDAE:

1) For each layer l of the SDAE network,
   a) For each column n of the weight matrix $\mathbf{W}_l$, draw
   $$\mathbf{W}_{l,*n} \sim \mathcal{N}(\mathbf{0}, \lambda_w^{-1}\mathbf{I}_{K_l}).$$
   b) Draw the bias vector $\mathbf{b}_l \sim \mathcal{N}(\mathbf{0}, \lambda_w^{-1}\mathbf{I}_{K_l})$.
   c) For each row j of $\mathbf{X}_l$, draw
   $$\mathbf{X}_{l,j*} \sim \mathcal{N}(\sigma(\mathbf{X}_{l-1,j*}\mathbf{W}_l + \mathbf{b}_l), \lambda_s^{-1}\mathbf{I}_{K_l}). \tag{7}$$
2) For each item j, draw a clean input [5]
   $$\mathbf{X}_{c,j*} \sim \mathcal{N}(\mathbf{X}_{L,j*}, \lambda_n^{-1}\mathbf{I}_B).$$

Note that if λ_s goes to infinity, the Gaussian distribution in Equation (7) will become a Dirac delta distribution [58] centered at $\sigma(\mathbf{X}_{l-1,j*}\mathbf{W}_l + \mathbf{b}_l)$, where $\sigma(\cdot)$ is the sigmoid function. The model will degenerate to be a Bayesian formulation of SDAE. That is why we call it generalized SDAE.

Note that the first $L/2$ layers of the network act as an encoder and the last $L/2$ layers act as a decoder. Maximization of the posterior probability is equivalent to minimization of the reconstruction error with weight decay taken into consideration.

Collaborative Deep Learning. Using the Bayesian SDAE as a component, the generative process of CDL is defined as follows:

1) Generate variables of generalized Bayesian SDAE.
2) For each item j,
   a) Draw the latent item offset vector $\epsilon_j \sim \mathcal{N}(\mathbf{0}, \lambda_v^{-1}\mathbf{I}_K)$ and then set the latent item vector: $\mathbf{v}_j = \epsilon_j + \mathbf{X}_{\frac{L}{2},j*}^T$.
3) Draw a latent user vector for each user i:
   $$\mathbf{u}_i \sim \mathcal{N}(\mathbf{0}, \lambda_u^{-1}\mathbf{I}_K).$$
4) Draw a rating $\mathbf{R}_{ij}$ for each user-item pair (i, j): $\mathbf{R}_{ij} \sim \mathcal{N}(\mathbf{u}_i^T\mathbf{v}_j, \mathbf{C}_{ij}^{-1})$.

Here λ_w, λ_n, λ_u, λ_s, and λ_v are hyperparameters and $\mathbf{C}_{ij}$ is a confidence parameter similar to that for CTR [60] ($\mathbf{C}_{ij} = a$ if $\mathbf{R}_{ij} = 1$ and $\mathbf{C}_{ij} = b$ otherwise). Note that the middle layer $\mathbf{X}_{L/2}$ serves as a bridge between the ratings and content information. This middle layer, along with the latent offset ϵ_j, is the key that enables CDL to simultaneously learn an effective feature representation and capture the similarity and (implicit) relationship between items (and users). Similar to the generalized SDAE, for computational efficiency, we can also take λ_s to infinity.

The graphical model of CDL when λ_s approaches positive infinity is shown in Fig. 4, where, for notational simplicity, we use $\mathbf{x}_0$, $\mathbf{x}_{L/2}$, and $\mathbf{x}_C$ in place of $\mathbf{X}_{0,j*}^T$, $\mathbf{X}_{\frac{L}{2},j*}^T$, and $\mathbf{X}_{c,j*}^T$, respectively.

Note that according the definition in Section 4.1, here the perception variables $\Omega_p = \{\{\mathbf{W}_l\}, \{\mathbf{b}_l\}, \{\mathbf{X}_l\}, \mathbf{X}_c\}$, the hinge variables $\Omega_h = \{\mathbf{V}\}$, and the task variables $\Omega_t = \{\mathbf{U}, \mathbf{R}\}$, where $\mathbf{V} = (\mathbf{v}_j)_{j=1}^J$ and $\mathbf{U} = (\mathbf{u}_i)_{i=1}^I$.

Learning. Based on the CDL model above, all parameters could be treated as random variables so that fully Bayesian

5. Note that while generation of the *clean* input $\mathbf{X}_c$ from $\mathbf{X}_L$ is part of the generative process of the Bayesian SDAE, generation of the *noise-corrupted* input $\mathbf{X}_0$ from $\mathbf{X}_c$ is an artificial noise injection process to help the SDAE learn a more robust feature representation.

Authorized licensed use limited to: Rutgers University Libraries. Downloaded on September 15,2024 at 02:31:03 UTC from IEEE Xplore. Restrictions apply.

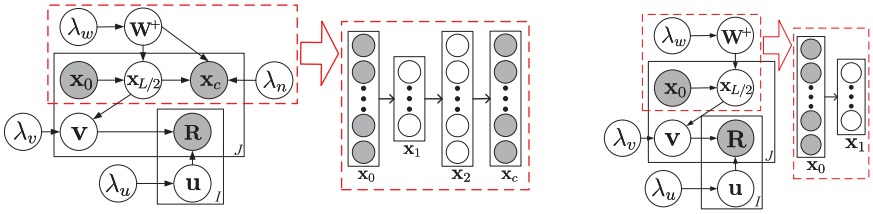


Fig. 4. On the left is the graphical model of CDL. The part inside the dashed rectangle represents an SDAE. An example SDAE with $L = 2$ is shown. On the right is the graphical model of the degenerated CDL. The part inside the dashed rectangle represents the encoder of an SDAE. An example SDAE with $L = 2$ is shown on its right. Note that although L is still 2, the decoder of the SDAE vanishes. To prevent clutter, we omit all variables $\mathbf{x}_l$ except $\mathbf{x}_0$ and $\mathbf{x}_{L/2}$ in the graphical models.

methods such as Markov chain Monte Carlo or variational approximation methods [32] may be applied. However, such treatment typically incurs high computational cost. Consequently, CDL uses an EM-style algorithm for obtaining the MAP estimates, as in [60].

As in CTR [60], maximizing the posterior probability is equivalent to maximizing the joint log-likelihood of $\mathbf{U}$, $\mathbf{V}$, $\{\mathbf{X}_l\}$, $\mathbf{X}_c$, $\{\mathbf{W}_l\}$, $\{\mathbf{b}_l\}$, and $\mathbf{R}$ given λ_u, λ_v, λ_w, λ_s, and λ_n

$$\begin{aligned}\mathscr{L} = &-\frac{\lambda_u}{2}\sum_i \|\mathbf{u}_i\|_2^2 - \frac{\lambda_w}{2}\sum_l (\|\mathbf{W}_l\|_F^2 + \|\mathbf{b}_l\|_2^2)\\ &-\frac{\lambda_v}{2}\sum_j \|\mathbf{v}_j - \mathbf{X}_{\frac{L}{2},j*}^T\|_2^2 - \frac{\lambda_n}{2}\sum_j \|\mathbf{X}_{L,j*} - \mathbf{X}_{c,j*}\|_2^2\\ &-\frac{\lambda_s}{2}\sum_l\sum_j \|\sigma(\mathbf{X}_{l-1,j*}\mathbf{W}_l + \mathbf{b}_l) - \mathbf{X}_{l,j*}\|_2^2\\ &-\sum_{i,j}\frac{\mathbf{C}_{ij}}{2}(\mathbf{R}_{ij} - \mathbf{u}_i^T\mathbf{v}_j)^2.\end{aligned}$$

If λ_s goes to infinity, the likelihood becomes:

$$\begin{aligned}\mathscr{L} = &-\frac{\lambda_u}{2}\sum_i \|\mathbf{u}_i\|_2^2 - \frac{\lambda_w}{2}\sum_l (\|\mathbf{W}_l\|_F^2 + \|\mathbf{b}_l\|_2^2)\\ &-\frac{\lambda_v}{2}\sum_j \|\mathbf{v}_j - f_e(\mathbf{X}_{0,j*}, \mathbf{W}^+)^T\|_2^2\\ &-\frac{\lambda_n}{2}\sum_j \|f_r(\mathbf{X}_{0,j*}, \mathbf{W}^+) - \mathbf{X}_{c,j*}\|_2^2\\ &-\sum_{i,j}\frac{\mathbf{C}_{ij}}{2}(\mathbf{R}_{ij} - \mathbf{u}_i^T\mathbf{v}_j)^2,\end{aligned} \tag{8}$$

where the encoder function $f_e(\cdot, \mathbf{W}^+)$ takes the corrupted content vector $\mathbf{X}_{0,j*}$ of item j as input and computes the encoding of the item, and the function $f_r(\cdot, \mathbf{W}^+)$ also takes $\mathbf{X}_{0,j*}$ as input, computes the encoding and then reconstructs the content vector of item j. For example, if the number of layers $L = 6$, $f_e(\mathbf{X}_{0,j*}, \mathbf{W}^+)$ is the output of the third layer while $f_r(\mathbf{X}_{0,j*}, \mathbf{W}^+)$ is the output of the sixth layer.

From the optimization perspective, the third term in the objective function (8) above is equivalent to a multi-layer perceptron using the latent item vectors $\mathbf{v}_j$ as the target while the fourth term is equivalent to an SDAE minimizing the reconstruction error. From the perspective of neural networks (NN), when λ_s approaches positive infinity, training of the probabilistic graphical model of CDL in Fig. 4(left) would degenerate to simultaneously training two neural networks overlaid together with a common input layer (the corrupted input) but different output layers, as shown in Fig. 5. Note that the second network is much more complex than typical neural networks due to the involvement of the rating matrix.

When the ratio λ_n/λ_v approaches positive infinity, it will degenerate to a two-step model in which the latent representation learned using SDAE is put directly into the CTR. The interaction between the perception component and the task-specific component is one-way (from the perception component to the task-specific component), meaning that the perception component will not be affected by the task-specific component. Another extreme happens when λ_n/λ_v goes to zero where the decoder of the SDAE essentially vanishes. On the right of Fig. 4 is the graphical model of the degenerated CDL when λ_n/λ_v goes to zero. As demonstrated in the experiments, the predictive performance will suffer greatly for both extreme cases [67]. This verifies that (1) the information from the task-specific component can improve the perception component, and (2) mutual boosting effect is crucial to BDL.

For $\mathbf{u}_i$ and $\mathbf{v}_j$, block coordinate descent similar to [30], [60] is used. Given the current $\mathbf{W}^+$, we compute the gradients of $\mathscr{L}$ with respect to $\mathbf{u}_i$ and $\mathbf{v}_j$ and then set them to zero, leading to the following update rules:

$$\begin{aligned}\mathbf{u}_i &\leftarrow (\mathbf{V}\mathbf{C}_i\mathbf{V}^T + \lambda_u\mathbf{I}_K)^{-1}\mathbf{V}\mathbf{C}_i\mathbf{R}_i\\ \mathbf{v}_j &\leftarrow (\mathbf{U}\mathbf{C}_i\mathbf{U}^T + \lambda_v\mathbf{I}_K)^{-1}(\mathbf{U}\mathbf{C}_j\mathbf{R}_j + \lambda_v f_e(\mathbf{X}_{0,j*}, \mathbf{W}^+)^T),\end{aligned}$$

where $\mathbf{U} = (\mathbf{u}_i)_{i=1}^I$, $\mathbf{V} = (\mathbf{v}_j)_{j=1}^J$, $\mathbf{C}_i = \mathrm{diag}(\mathbf{C}_{i1}, \ldots, \mathbf{C}_{iJ})$ is a diagonal matrix, $\mathbf{R}_i = (\mathbf{R}_{i1}, \ldots, \mathbf{R}_{iJ})^T$ is a column vector containing all the ratings of user i, and $\mathbf{C}_{ij}$ reflects the confidence controlled by a and b as discussed in [30]. $\mathbf{C}_j$ and $\mathbf{R}_j$ are defined similarly for item j.

Given $\mathbf{U}$ and $\mathbf{V}$, we can learn the weights $\mathbf{W}_l$ and biases $\mathbf{b}_l$ for each layer using the back-propagation (BP) learning algorithm. The gradients of the likelihood with respect to $\mathbf{W}_l$ and $\mathbf{b}_l$ are as follows:

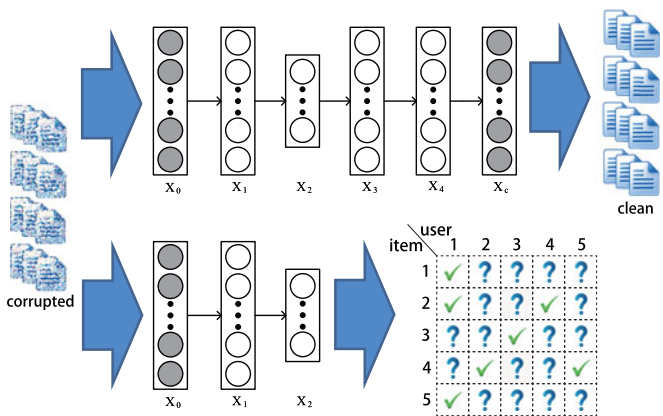


Fig. 5. NN representation for degenerated CDL.

Authorized licensed use limited to: Rutgers University Libraries. Downloaded on September 15,2024 at 02:31:03 UTC from IEEE Xplore. Restrictions apply.

TABLE 2
Recall@300 on the Dataset *Citeulike-a* (%)

SVDFeature [14]	CMF [56]	DeepMusic [45]	CTR [60]	CDL [67]
11.19	13.45	12.32	22.11	**31.06**

$$\begin{aligned}\nabla_{\mathbf{W}_l}\mathscr{L} &= -\lambda_w \mathbf{W}_l \\ &\quad - \lambda_v \sum_j \nabla_{\mathbf{W}_l} f_e(\mathbf{X}_{0,j*}, \mathbf{W}^+)^T (f_e(\mathbf{X}_{0,j*}, \mathbf{W}^+)^T - \mathbf{v}_j) \\ &\quad - \lambda_n \sum_j \nabla_{\mathbf{W}_l} f_r(\mathbf{X}_{0,j*}, \mathbf{W}^+)(f_r(\mathbf{X}_{0,j*}, \mathbf{W}^+) - \mathbf{X}_{c,j*}) \\ \nabla_{\mathbf{b}_l}\mathscr{L} &= -\lambda_w \mathbf{b}_l \\ &\quad - \lambda_v \sum_j \nabla_{\mathbf{b}_l} f_e(\mathbf{X}_{0,j*}, \mathbf{W}^+)^T (f_e(\mathbf{X}_{0,j*}, \mathbf{W}^+)^T - \mathbf{v}_j) \\ &\quad - \lambda_n \sum_j \nabla_{\mathbf{b}_l} f_r(\mathbf{X}_{0,j*}, \mathbf{W}^+)(f_r(\mathbf{X}_{0,j*}, \mathbf{W}^+) - \mathbf{X}_{c,j*}).\end{aligned}$$

By alternating the update of $\mathbf{U}$, $\mathbf{V}$, $\mathbf{W}_l$, and $\mathbf{b}_l$, we can find a local optimum for $\mathscr{L}$. Several commonly used techniques such as using a momentum term may be applied to alleviate the local optimum problem. Note that a carefully designed BDL model (according to the i.i.d. requirement and with proper variance models as stated in Section 4.1) can minimize the overhead of seamlessly combining the perception component and the task-specific component. In CDL, the computational complexity (per iteration) of the perception component is $O(JBK_1)$ and that of the task-specific component is $O(K^2 N_R + K^3)$, where N_R is the number of non-zero entries in the rating matrix and $K = K_{\frac{L}{2}}$. The computational complexity (per iteration) for the whole model is $O(JBK_1 + K^2 N_R + K^3)$ [67]. No significant overhead is introduced.

Prediction. Let D be the observed test data. Similar to [60], CDL uses the point estimates of $\mathbf{u}_i$, $\mathbf{W}^+$ and ϵ_j to calculate the predicted rating

$$E[\mathbf{R}_{ij}|D] \approx E[\mathbf{u}_i|D]^T (E[f_e(\mathbf{X}_{0,j*}, \mathbf{W}^+)^T|D] + E[\epsilon_j|D]),$$

where $E[\cdot]$ denotes the expectation operation. In other words, we approximate the predicted rating as

$$\mathbf{R}^*_{ij} \approx (\mathbf{u}^*_j)^T (f_e(\mathbf{X}_{0,j*}, \mathbf{W}^{+*})^T + \epsilon^*_j) = (\mathbf{u}^*_i)^T \mathbf{v}^*_j.$$

Note that for any new item j with no rating in the training data, its offset ϵ^*_j will be $\mathbf{0}$.

Table 2 shows the recall of recommendation with 300 recommended items for different methods in the dataset *citeulike-a*. Please refer to [67] for more details.

In the following text, we provide several extensions of CDL from different perspectives.

4.2.2 Bayesian Collaborative Deep Learning

Besides the MAP estimates, a sampling-based algorithm for the Bayesian treatment of CDL is also proposed in [67]. This algorithm turns out to be a Bayesian and generalized version of the well-known back-propagation learning algorithm. We list the key conditional densities as follows:

For $\mathbf{W}+$. We denote the concatenation of $\mathbf{W}_{l,*n}$ and $\mathbf{b}_l^{(n)}$ as $\mathbf{W}^+_{l,*n}$. Similarly, the concatenation of $\mathbf{X}_{l,j*}$ and 1 is denoted as $\mathbf{X}^+_{l,j*}$. The subscripts of $\mathbf{I}$ are ignored. Then

$$\begin{aligned}&p(\mathbf{W}^+_{l,*n}|\mathbf{X}_{l-1,j*}, \mathbf{X}_{l,j*}, \lambda_s) \\ &\propto \mathcal{N}(\mathbf{W}^+_{l,*n}|0, \lambda_w^{-1}\mathbf{I})\mathcal{N}(\mathbf{X}_{l,*n}|\sigma(\mathbf{X}^+_{l-1}\mathbf{W}^+_{l,*n}), \lambda_s^{-1}\mathbf{I}).\end{aligned}$$

For $\mathbf{X}_{l,j*}$ $(l \neq L/2)$. Similarly, we denote the concatenation of $\mathbf{W}_l$ and $\mathbf{b}_l$ as $\mathbf{W}^+_l$ and have

$$\begin{aligned}&p(\mathbf{X}_{l,j*}|\mathbf{W}^+_l, \mathbf{W}^+_{l+1}, \mathbf{X}_{l-1,j*}, \mathbf{X}_{l+1,j*}\lambda_s) \\ &\propto \mathcal{N}(\mathbf{X}_{l,j*}|\sigma(\mathbf{X}^+_{l-1,j*}\mathbf{W}^+_l), \lambda_s^{-1}\mathbf{I})\cdot \\ &\quad \mathcal{N}(\mathbf{X}_{l+1,j*}|\sigma(\mathbf{X}^+_{l,j*}\mathbf{W}^+_{l+1}), \lambda_s^{-1}\mathbf{I}).\end{aligned}$$

Note that for the last layer $(l = L)$ the second Gaussian would be $\mathcal{N}(\mathbf{X}_{c,j*}|\mathbf{X}_{l,j*}, \lambda_s^{-1}\mathbf{I})$ instead.

For $\mathbf{X}_{l,j*}$ $(l = L/2)$. Similarly, we have

$$\begin{aligned}&p(\mathbf{X}_{l,j*}|\mathbf{W}^+_l, \mathbf{W}^+_{l+1}, \mathbf{X}_{l-1,j*}, \mathbf{X}_{l+1,j*}, \lambda_s, \lambda_v, \mathbf{v}_j) \\ &\propto \mathcal{N}(\mathbf{X}_{l,j*}|\sigma(\mathbf{X}^+_{l-1,j*}\mathbf{W}^+_l), \lambda_s^{-1}\mathbf{I})\cdot \\ &\quad \mathcal{N}(\mathbf{X}_{l+1,j*}|\sigma(\mathbf{X}^+_{l,j*}\mathbf{W}^+_{l+1}), \lambda_s^{-1}\mathbf{I})\mathcal{N}(\mathbf{v}_j|\mathbf{X}_{l,j*}, \lambda_v^{-1}\mathbf{I}).\end{aligned}$$

For $\mathbf{v}_j$. The posterior $p(\mathbf{v}_j|\mathbf{X}_{L/2,j*}, \mathbf{R}_{*j}, \mathbf{C}_{*j}, \lambda_v, \mathbf{U})$

$$\propto \mathcal{N}(\mathbf{v}_j|\mathbf{X}^T_{L/2,j*}, \lambda_v^{-1}\mathbf{I}) \prod_i \mathcal{N}(\mathbf{R}_{ij}|\mathbf{u}_i^T\mathbf{v}_j, \mathbf{C}_{ij}^{-1}).$$

For $\mathbf{u}_i$. The posterior $p(\mathbf{u}_i|\mathbf{R}_{i*}, \mathbf{V}, \lambda_u, \mathbf{C}_{i*})$

$$\propto \mathcal{N}(\mathbf{u}_i|0, \lambda_u^{-1}\mathbf{I}) \prod_j (\mathbf{R}_{ij}|\mathbf{u}_i^T\mathbf{v}_j|\mathbf{C}_{ij}^{-1}).$$

Interestingly, if λ_s goes to infinity and adaptive rejection Metropolis sampling (which involves using the gradients of the objective function to approximate the proposal distribution) is used, the sampling for $\mathbf{W}^+$ turns out to be a *Bayesian generalized* version of BP. Specifically, as Fig. 6 shows, after getting the gradient of the loss function at one point (the red dashed line on the left), the next sample would be drawn in the region under that line, which is equivalent to a probabilistic version of BP. If a sample is above the curve of the loss function, a new tangent line (the black dashed line on the right) would be added to better approximate the distribution corresponding to the joint log-likelihood. After that, samples would be drawn from the region under both lines. During the sampling, besides searching for local optima using the gradients (MAP), the algorithm also takes the variance into consideration. That is why it is called *Bayesian generalized back-propagation*.

4.2.3 Marginalized Collaborative Deep Learning

In SDAE, corrupted input goes through encoding and decoding to recover the clean input. Usually, different epochs of training use different corrupted versions as input. Hence generally, SDAE needs to go through enough epochs of training to see sufficient corrupted versions of the input. Marginalized SDAE (mSDAE) [13] seeks to avoid this by marginalizing out the corrupted input and obtaining closed-form solutions directly. In this sense, mSDAE is more computationally efficient than SDAE.

Authorized licensed use limited to: Rutgers University Libraries. Downloaded on September 15,2024 at 02:31:03 UTC from IEEE Xplore. Restrictions apply.

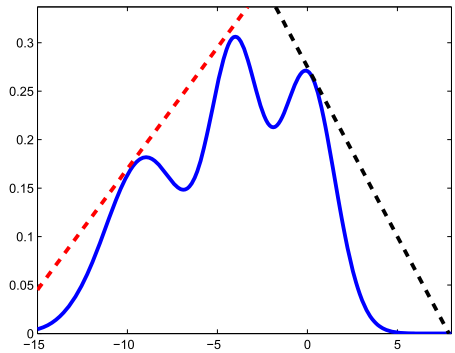


Fig. 6. Sampling as generalized BP.

As mentioned in [37], using mSDAE instead of the Bayesian SDAE could lead to more efficient learning algorithms. For example, in [37], the objective when using a one-layer mSDAE can be written as follows:

$$\begin{aligned}\mathscr{L} = &-\sum_j \|\widetilde{\mathbf{X}}_{0,j*}\mathbf{W}_1 - \overline{\mathbf{X}}_{c,j*}\|_2^2 - \sum_{i,j}\frac{\mathbf{C}_{ij}}{2}(\mathbf{R}_{ij} - \mathbf{u}_i^T\mathbf{v}_j)^2 \\ &-\frac{\lambda_u}{2}\sum_i \|\mathbf{u}_i\|_2^2 - \frac{\lambda_v}{2}\sum_j \|\mathbf{v}_j^T\mathbf{P}_1 - \mathbf{X}_{0,j*}\mathbf{W}_1\|_2^2,\end{aligned} \tag{9}$$

where $\widetilde{\mathbf{X}}_{0,j*}$ is the collection of k different corrupted versions of $\mathbf{X}_{0,j*}$ (a k-by-B matrix) and $\overline{\mathbf{X}}_{c,j*}$ is the k-time repeated version of $\mathbf{X}_{c,j*}$ (also a k-by-B matrix). $\mathbf{P}_1$ is the transformation matrix for item latent factors.

The solution for $\mathbf{W}_1$ would be:

$$\mathbf{W}_1 = E(\mathbf{S}_1)E(\mathbf{Q}_1)^{-1},$$

where $\mathbf{S}_1 = \overline{\mathbf{X}}_{c,j*}^T\widetilde{\mathbf{X}}_{0,j*} + \frac{\lambda_u}{2}\mathbf{P}_1^T\mathbf{V}\mathbf{X}_c$ and $\mathbf{Q}_1 = \overline{\mathbf{X}}_{c,j*}^T\widetilde{\mathbf{X}}_{0,j*} + \frac{\lambda_u}{2}\mathbf{X}_c^T\mathbf{X}_c$. A solver for the expectation in the equation above is provided in [13]. Note that this is a linear and one-layer case which can be generalized to the nonlinear and multi-layer case using the same techniques as in [12], [13].

As we can see, in marginalized CDL, the perception variables $\Omega_p = \{\mathbf{X}_0, \mathbf{X}_c, \mathbf{W}_1\}$, the hinge variables $\Omega_h = \{\mathbf{V}\}$, and the task variables $\Omega_t = \{\mathbf{P}_1, \mathbf{R}, \mathbf{U}\}$.

4.2.4 Collaborative Deep Ranking

CDL assumes a collaborative filtering setting to model the ratings directly. However, the output of recommender systems is often a ranked list, which means it would be more natural to use ranking rather than ratings as the objective. With this motivation, collaborative deep ranking (CDR) is proposed [71] to jointly perform representation learning and collaborative ranking. The corresponding generative process is the same as that of CDL except for Step 3 and 4, which should be replaced with:

- For each user i,
  1) Draw a latent user vector for each user i:
  $$\mathbf{u}_i \sim \mathcal{N}(\mathbf{0}, \lambda_u^{-1}\mathbf{I}_K).$$
  2) For each pair-wise preference $(j,k) \in \mathcal{P}_i$, where $\mathcal{P}_i = \{(j,k) : \mathbf{R}_{ij} - \mathbf{R}_{ik} > 0\}$, draw the preference: $\Delta_{ijk} \sim \mathcal{N}(\mathbf{u}_i^T\mathbf{v}_j - \mathbf{u}_i^T\mathbf{v}_k, \mathbf{C}_{ijk}^{-1})$.

Following the generative process, the last term of Equation (8) becomes $-\sum_{i,j,k}\frac{\mathbf{C}_{ijk}}{2}(\Delta_{ijk} - (\mathbf{u}_i^T\mathbf{v}_j - \mathbf{u}_i^T\mathbf{v}_k))^2$. Similar algorithms can be used to learn the parameters in CDR. As reported in [71], using the ranking objective leads to significant improvement in the recommendation performance.

Following the definition in Section 4.1, CDR's perception variables $\Omega_p = \{\{\mathbf{W}_l\}, \{\mathbf{b}_l\}, \{\mathbf{X}_l\}, \mathbf{X}_c\}$, the hinge variables $\Omega_h = \{\mathbf{V}\}$, and the task variables $\Omega_t = \{\mathbf{U}, \Delta\}$.

4.2.5 Symmetric Collaborative Deep Learning

Models like [67], [71] focus the deep learning component on modeling the item content. Besides the content information from the items, attributes of users sometimes contain much more important information. It is therefore desirable to extend CDL to model user attributes as well [37]. We call this variant symmetric CDL. For example, using an extra mSDAE on the user attributes adds two extra terms in Equation (9), $-\frac{\lambda_u}{2}\sum_i \|\mathbf{u}_i^T\mathbf{P}_2 - \mathbf{Y}_{0,j*}\mathbf{W}_2\|_2^2$ and $-\sum_i \|\widetilde{\mathbf{Y}}_{0,i*}\mathbf{W}_2 - \overline{\mathbf{Y}}_{c,i*}\|_2^2$, where $\widetilde{\mathbf{Y}}_{0,j*}$ (a k-by-D matrix for user attributes) is the collection of k different corrupted versions of $\mathbf{Y}_{0,j*}$ and $\overline{\mathbf{Y}}_{c,i*}$ (also a k-by-D matrix) is the k-time repeated version of $\mathbf{Y}_{c,i*}$ (the clean user attributes). $\mathbf{P}_2$ is the transformation matrix for user latent factors and D is the number of user attributes. Similar to the marginalized CDL, the solution for $\mathbf{W}_2$ given other parameters is

$$\mathbf{W}_2 = E(\mathbf{S}_2)E(\mathbf{Q}_2)^{-1},$$

where $\mathbf{S}_2 = \overline{\mathbf{Y}}_{c,i*}^T\widetilde{\mathbf{Y}}_{0,i*} + \frac{\lambda_u}{2}\mathbf{P}_2^T\mathbf{U}\mathbf{Y}_c$ and $\mathbf{Q}_2 = \overline{\mathbf{Y}}_{c,i*}^T\widetilde{\mathbf{Y}}_{0,i*} + \frac{\lambda_u}{2}\mathbf{Y}_c^T\mathbf{Y}_c$.

In symmetric CDL, the perception variables $\Omega_p = \{\mathbf{X}_0, \mathbf{X}_c, \mathbf{W}_1, \mathbf{Y}_0, \mathbf{Y}_c, \mathbf{W}_2\}$, the hinge variables $\Omega_h = \{\mathbf{V}, \mathbf{U}\}$, and the task variables $\Omega_t = \{\mathbf{P}_1, \mathbf{P}_2, \mathbf{R}\}$.

4.2.6 Discussion

CDL is the first hierarchical Bayesian model to bridge the gap between state-of-the-art deep learning models and RS. By performing deep learning collaboratively, CDL and its variants can simultaneously extract an effective deep feature representation from the content and capture the similarity and implicit relationship between items (and users). This way, the perception component and the task-specific component are able to interact with each other to create synergy and further boost the recommendation accuracy. The learned representation may also be used for tasks other than recommendation. Unlike previous deep learning models which use a simple target such as classification [33] and reconstruction [59], CDL-based models[6] use CF as a more complex target in a probabilistic framework.

As mentioned in Section 1, the synergy created by *information exchange* between two components is crucial to the performance of BDL. In the CDL-based models above, the exchange is achieved by assuming Gaussian distributions that connect the hinge variables and the variables in the perception component (drawing the hinge variable $\mathbf{v}_j \sim \mathcal{N}(\mathbf{X}_{\frac{L}{2},j*}^T, \lambda_v^{-1}\mathbf{I}_K)$ in the generative process of CDL, where $\mathbf{X}_{\frac{L}{2}}$ is a perception variable), which is simple but effective and efficient in computation. Among the five CDL-

6. During the review process of this paper, there are some newly published works based on BDL (e.g., some CDL-based works [65], [72]).

Authorized licensed use limited to: Rutgers University Libraries. Downloaded on September 15,2024 at 02:31:03 UTC from IEEE Xplore. Restrictions apply.

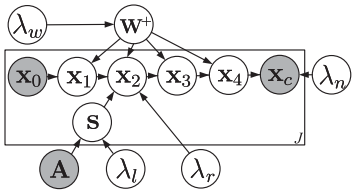


Fig. 7. Graphical model of RSDAE for $L = 4$. λ_s is not shown here to prevent clutter.

based models in Table 1, three of them are HV models and the others are LV models, according to the definition in Section 4.1. Since it has been verified that the HV CDL significantly outperforms its ZV counterpart [67], we can expect extra performance boosts from the LV counterparts of the three HV models.

Besides efficient *information exchange*, the designs of the models also meet the i.i.d. requirement of the distribution concerning hinge variables discussed in Section 4.1 and are hence easily parallelizable. In some models to be introduced later, we will see alternative designs to enable efficient and i.i.d. information exchange between the two components of BDL.

4.3 Bayesian Deep Learning for Topic Models

In this section, we review some examples of using BDL for topic models. These models combine the merits of PGM (which naturally incorporates the probabilistic relationships among variables) and NN (which learns deep representations efficiently), leading to significant performance boost.

4.3.1 Relational Stacked Denoising Autoencoders as Topic Models

Problem Statement and Notation. Assume we have a set of items (articles or movies) $\mathbf{X}_c$, with $\mathbf{X}_{c,j*}^T \in \mathbb{R}^B$ denoting the content (attributes) of item j. Besides, we use $\mathbf{I}_K$ to denote a K-dimensional identity matrix and $\mathbf{S} = [\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_J]$ to denote the *relational latent matrix* with $\mathbf{s}_j$ representing the relational properties of item j.

From the perspective of SDAE, the J-by-B matrix $\mathbf{X}_c$ represents the clean input to the SDAE and the noise-corrupted matrix of the same size is denoted by $\mathbf{X}_0$. Besides, we denote the output of layer l of the SDAE, a J-by-K_l matrix, by $\mathbf{X}_l$. Row j of $\mathbf{X}_l$ is denoted by $\mathbf{X}_{l,j*}$, $\mathbf{W}_l$ and $\mathbf{b}_l$ are the weight matrix and bias vector of layer l, $\mathbf{W}_{l,*n}$ denotes column n of $\mathbf{W}_l$, and L is the number of layers. As a shorthand, we refer to the collection of weight matrices and biases in all layers as $\mathbf{W}^+$. Note that an $L/2$-layer SDAE corresponds to an L-layer network.

Model Formulation. Here we will use the Bayesian SDAE introduced before as a building block for the relational stacked denoising autoencoder (RSDAE) model.

As mentioned in [64], RSDAE is formulated as a novel probabilistic model which can seamlessly integrate layered representation learning and the relational information available. This way, the model can simultaneously learn the feature representation from the content information and the relation between items. The graphical model of RSDAE is shown in Fig. 7 and the generative process is listed as follows:

1) Draw the relational latent matrix $\mathbf{S}$ from a *matrix variate normal distribution* [21]:

$$\mathbf{S} \sim \mathcal{N}_{K,J}(0, \mathbf{I}_K \otimes (\lambda_l \mathscr{L}_a)^{-1}). \tag{10}$$

2) For layer l of the SDAE where $l = 1, 2, \ldots, \frac{L}{2} - 1$,
   a) For each column n of the weight matrix $\mathbf{W}_l$, draw $\mathbf{W}_{l,*n} \sim \mathcal{N}(0, \lambda_w^{-1}\mathbf{I}_{K_l})$.
   b) Draw the bias vector $\mathbf{b}_l \sim \mathcal{N}(0, \lambda_w^{-1}\mathbf{I}_{K_l})$.
   c) For each row j of $\mathbf{X}_l$, draw
$$\mathbf{X}_{l,j*} \sim \mathcal{N}(\sigma(\mathbf{X}_{l-1,j*}\mathbf{W}_l + \mathbf{b}_l), \lambda_s^{-1}\mathbf{I}_{K_l}).$$
3) For layer $\frac{L}{2}$ of the SDAE network, draw the representation vector for item j from the product of two Gaussians (PoG) [16]:
$$\mathbf{X}_{\frac{L}{2},j*} \sim \mathrm{PoG}(\sigma(\mathbf{X}_{\frac{L}{2}-1,j*}\mathbf{W}_l + \mathbf{b}_l), \mathbf{s}_j^T, \lambda_s^{-1}\mathbf{I}_K, \lambda_r^{-1}\mathbf{I}_K). \tag{11}$$
4) For layer l of the SDAE network where $l = \frac{L}{2} + 1, \frac{L}{2} + 2, \ldots, L$,
   a) For each column n of the weight matrix $\mathbf{W}_l$, draw $\mathbf{W}_{l,*n} \sim \mathcal{N}(0, \lambda_w^{-1}\mathbf{I}_{K_l})$.
   b) Draw the bias vector $\mathbf{b}_l \sim \mathcal{N}(0, \lambda_w^{-1}\mathbf{I}_{K_l})$.
   c) For each row j of $\mathbf{X}_l$, draw
$$\mathbf{X}_{l,j*} \sim \mathcal{N}(\sigma(\mathbf{X}_{l-1,j*}\mathbf{W}_l + \mathbf{b}_l), \lambda_s^{-1}\mathbf{I}_{K_l}).$$
5) For each item j, draw a clean input
$$\mathbf{X}_{c,j*} \sim \mathcal{N}(\mathbf{X}_{L,j*}, \lambda_n^{-1}\mathbf{I}_B).$$

Here $K = K_{\frac{L}{2}}$ is the dimensionality of the learned representation vector for each item, $\mathbf{S}$ denotes the $K \times J$ relational latent matrix in which column j is the *relational latent vector* $\mathbf{s}_j$ for item j. Note that $\mathcal{N}_{K,J}(0, \mathbf{I}_K \otimes (\lambda_l \mathscr{L}_a)^{-1})$ in Equation (10) is a matrix variate normal distribution defined as in [21]

$$\begin{aligned} p(\mathbf{S}) &= \mathcal{N}_{K,J}(0, \mathbf{I}_K \otimes (\lambda_l \mathscr{L}_a)^{-1}) \\ &= \frac{\exp\{\mathrm{tr}[-\frac{\lambda_l}{2}\mathbf{S}\mathscr{L}_a\mathbf{S}^T]\}}{(2\pi)^{JK/2}|\mathbf{I}_K|^{J/2}|\lambda_l \mathscr{L}_a|^{-K/2}}, \end{aligned} \tag{12}$$

where the operator $\otimes$ denotes the Kronecker product of two matrices [21], $\mathrm{tr}(\cdot)$ denotes the trace of a matrix, and $\mathscr{L}_a$ is the Laplacian matrix incorporating the relational information. $\mathscr{L}_a = \mathbf{D} - \mathbf{A}$, where $\mathbf{D}$ is a diagonal matrix whose diagonal elements $\mathbf{D}_{ii} = \sum_j \mathbf{A}_{ij}$ and $\mathbf{A}$ is the adjacency matrix representing the relational information with binary entries indicating the links (or relations) between items. $\mathbf{A}_{jj'} = 1$ indicates that there is a link between item j and item j' and $\mathbf{A}_{jj'} = 0$ otherwise. $\mathrm{PoG}(\sigma(\mathbf{X}_{\frac{L}{2}-1,j*}\mathbf{W}_l + \mathbf{b}_l), \mathbf{s}_j^T, \lambda_s^{-1}\mathbf{I}_K, \lambda_r^{-1}\mathbf{I}_K)$ denotes the product of the Gaussian $\mathcal{N}(\sigma(\mathbf{X}_{\frac{L}{2}-1,j*}\mathbf{W}_l + \mathbf{b}_l), \lambda_s^{-1}\mathbf{I}_K)$ and the Gaussian $\mathcal{N}(\mathbf{s}_j^T, \lambda_r^{-1}\mathbf{I}_K)$, which is also a Gaussian [16].

According to the generative process above, maximizing the posterior probability is equivalent to maximizing the joint log-likelihood of $\{\mathbf{X}_l\}$, $\mathbf{X}_c$, $\mathbf{S}$, $\{\mathbf{W}_l\}$, and $\{\mathbf{b}_l\}$ given λ_s, λ_w, λ_l, λ_r, and λ_n

$$\begin{aligned} \mathscr{L} = &-\frac{\lambda_l}{2}\mathrm{tr}(\mathbf{S}\mathscr{L}_a\mathbf{S}^T) - \frac{\lambda_r}{2}\sum_j \|(\mathbf{s}_j^T - \mathbf{X}_{\frac{L}{2},j*})\|_2^2 \\ &- \frac{\lambda_w}{2}\sum_l(\|\mathbf{W}_l\|_F^2 + \|\mathbf{b}_l\|_2^2) \\ &- \frac{\lambda_n}{2}\sum_j \|\mathbf{X}_{L,j*} - \mathbf{X}_{c,j*}\|_2^2 \\ &- \frac{\lambda_s}{2}\sum_l\sum_j \|\sigma(\mathbf{X}_{l-1,j*}\mathbf{W}_l + \mathbf{b}_l) - \mathbf{X}_{l,j*}\|_2^2, \end{aligned}$$

Authorized licensed use limited to: Rutgers University Libraries. Downloaded on September 15,2024 at 02:31:03 UTC from IEEE Xplore. Restrictions apply.

TABLE 3
Recall@300 on the Dataset *Movielens-Plot* (%)

CTR [60]	CTR-SR [62]	SDAE [59]	RSDAE [64]
20.43	23.07	23.45	**24.86**

where $\mathbf{X}_{l,j*} = \sigma(\mathbf{X}_{l-1,j*}\mathbf{W}_l + \mathbf{b}_l)$. Note that the first term $-\frac{\lambda_l}{2}\mathrm{tr}(\mathbf{S}\mathscr{L}_a\mathbf{S}^T)$ corresponds to $\log p(\mathbf{S})$ in the matrix variate distribution in Equation (12). By simple manipulation, we have $\mathrm{tr}(\mathbf{S}\mathscr{L}_a\mathbf{S}^T) = \sum_{k=1}^{K}\mathbf{S}_{k*}^T\mathscr{L}_a\mathbf{S}_{k*}$, where $\mathbf{S}_{k*}$ denotes the kth row of $\mathbf{S}$. As we can see, maximizing $-\frac{\lambda_l}{2}\mathrm{tr}(\mathbf{S}^T\mathscr{L}_a\mathbf{S})$ is equivalent to making $\mathbf{s}_j$ closer to $\mathbf{s}_{j'}$ if item j and item j' are linked (namely $\mathbf{A}_{jj'} = 1$).

In RSDAE, the perception variables $\Omega_p = \{\{\mathbf{X}_l\}, \mathbf{X}_c, \{\mathbf{W}_l\}, \{\mathbf{b}_l\}\}$, the hinge variables $\Omega_h = \{\mathbf{S}\}$, and the task variables $\Omega_t = \{\mathbf{A}\}$.

Learning Relational Representation and Topics. [64] provides an EM-style algorithm for MAP estimation. Here we review some of the key steps as follows.

In terms of the relational latent matrix $\mathbf{S}$, we first fix all rows of $\mathbf{S}$ except the kth one $\mathbf{S}_{k*}$ and then update $\mathbf{S}_{k*}$. Specifically, we take the gradient of $\mathscr{L}$ with respect to $\mathbf{S}_{k*}$, set it to 0, and get the following linear system:

$$(\lambda_l\mathscr{L}_a + \lambda_r\mathbf{I}_J)\mathbf{S}_{k*} = \lambda_r\mathbf{X}_{\frac{L}{2},*k}^T. \tag{13}$$

A naive approach is to solve the linear system by setting $\mathbf{S}_{k*} = \lambda_r(\lambda_l\mathscr{L}_a + \lambda_r\mathbf{I}_J)^{-1}\mathbf{X}_{\frac{L}{2},*k}^T$. Unfortunately, the complexity is $O(J^3)$ for one single update. Similar to [38], the steepest descent method [55] is used to iteratively update $\mathbf{S}_{k*}$

$$\begin{aligned}\mathbf{S}_{k*}(t+1) &\leftarrow \mathbf{S}_{k*}(t) + \delta(t)r(t)\\ r(t) &\leftarrow \lambda_r\mathbf{X}_{\frac{L}{2},*k}^T - (\lambda_l\mathscr{L}_a + \lambda_r\mathbf{I}_J)\mathbf{S}_{k*}(t)\\ \delta(t) &\leftarrow \frac{r(t)^T r(t)}{r(t)^T(\lambda_l\mathscr{L}_a + \lambda_r\mathbf{I}_J)r(t)}.\end{aligned}$$

As discussed in [38], the use of steepest descent method dramatically reduces the computation cost in each iteration from $O(J^3)$ to $O(J)$.

Given $\mathbf{S}$, we can learn $\mathbf{W}_l$ and $\mathbf{b}_l$ for each layer using the back-propagation algorithm. By alternating the update of $\mathbf{S}$, $\mathbf{W}_l$, and $\mathbf{b}_l$, a local optimum for $\mathscr{L}$ can be found. Also, techniques such as including a momentum term may help to avoid being trapped in a local optimum. The computational complexity for each iteration is $O(JBK_1 + KJ)$. Similar to CDL, no significant overhead is introduced.

Table 3 shows the recall for different methods in the dataset *movielens-plot* when the learned representation is used for tag recommendation (with 300 recommended tags for each item). As we can see, RSDAE significantly outperforms SDAE, which means that the relational information from the task-specific component is crucial to the performance boost. Please refer to [64] for more details.

4.3.2 Deep Poisson Factor Analysis with Sigmoid Belief Networks

The Poisson distribution with support over nonnegative integers is known as a natural choice to model counts. It is, therefore, desirable to use it as a building block for topic models [8]. With this motivation, [75] proposed a model, dubbed Poisson factor analysis (PFA), for latent nonnegative matrix factorization via Poisson distributions.

Poisson Factor Analysis. PFA assumes a discrete N-by-P matrix $\mathbf{X}$ containing word counts of N documents with a vocabulary size of P [17], [75]. In a nutshell, PFA can be described using the following equation:

$$\mathbf{X} \sim \mathrm{Pois}((\Theta \circ \mathbf{H})\Phi), \tag{14}$$

where Φ (of size K-by-P where K is the number of topics) denotes the factor loading matrix in factor analysis with the kth row ϕ_k encoding the importance of each word in topic k. The N-by-K matrix Θ is the factor score matrix with the nth row θ_n containing topic proportions for document n. The N-by-K matrix $\mathbf{H}$ is a latent binary matrix with the nth row $\mathbf{h}_n$ defining a set of topics associated with document n.

Different priors correspond to different models. For example, Dirichlet priors on ϕ_k and θ_n with an all-one matrix $\mathbf{H}$ would recover LDA [8] while a beta-Bernoulli prior on $\mathbf{h}_n$ leads to the negative binomial focused topic model (NB-FTM) model in [74]. In [17], a deep-structured prior based on sigmoid belief networks (SBN) [43] (an MLP variant with binary hidden units) is imposed on $\mathbf{h}_n$ to form a deep PFA model for topic modeling.

Deep Poisson Factor Analysis. In the deep PFA model [17], the generative process can be summarized as follows:

$$\begin{aligned}&\phi_k \sim \mathrm{Dir}(a_\phi, \ldots, a_\phi), \theta_{nk} \sim \mathrm{Gamma}\left(r_k, \frac{p_n}{1-p_n}\right)\\ &r_k \sim \mathrm{Gamma}\left(\gamma_0, \frac{1}{c_0}\right), \gamma_0 \sim \mathrm{Gamma}\left(e_0, \frac{1}{f_0}\right)\\ &h_{nk_L}^{(L)} \sim \mathrm{Ber}(\sigma(b_{k_L}^{(L)}))\end{aligned} \tag{15}$$

$$h_{nk_l}^{(l)} \sim \mathrm{Ber}(\sigma(\mathbf{h}_n^{(l+1)}\mathbf{w}_{k_l}^{(l)} + b_{k_l}^{(l)})) \tag{16}$$

$$x_{npk} \sim \mathrm{Pois}(\phi_{kp}\theta_{nk}h_{nk}^{(1)}), x_{np} = \sum_{k=1}^{K} x_{npk}, \tag{17}$$

where L is the number of layers in SBN, which corresponds to Equations (15) and (16). x_{np} is an entry in the matrix $\mathbf{X}$, $\mathbf{h}_n^{(l)}$ is the nth row of $\mathbf{H}_l$, and x_{npk} is the count of word p that comes from topic k in document n.

In this model, the perception variables $\Omega_p = \{\{\mathbf{H}_l\}, \{\mathbf{W}_l\}, \{\mathbf{b}_l\}\}$, the hinge variables $\Omega_h = \{\mathbf{X}\}$, and the task variables $\Omega_t = \{\{\phi_k\}, \{r_k\}, \Theta, \gamma_0\}$. $\mathbf{W}_l$ is the weight matrix containing columns of $\mathbf{w}_{k_l}^{(l)}$ and $\mathbf{b}_l$ is the bias vector containing entries of $b_{k_l}^{(l)}$ in Equation (16).

Learning Using Bayesian Conditional Density Filtering. Efficient learning algorithms are needed for Bayesian treatments of deep PFA. Gan et al. [17] proposed to use an online version of MCMC called Bayesian conditional density filtering (BCDF) to learn both the global parameters $\Psi_g = (\{\phi_k\}, \{r_k\}, \gamma_0, \{\mathbf{W}_l\}, \{\mathbf{b}_l\})$ and the local variables $\Psi_l = (\Theta, \{\mathbf{H}_l\})$. The key conditional densities used for the Gibbs updates are as follows:

$$\begin{aligned}x_{npk}|- &\sim \mathrm{Multi}(x_{np}; \zeta_{np1}, \ldots, \zeta_{npK})\\ \phi_k|- &\sim \mathrm{Dir}(a_\phi + x_{\cdot 1k}, \ldots, a_\phi + x_{\cdot Pk})\\ \theta_{nk}|- &\sim \mathrm{Gamma}(r_k h_{nk}^{(1)} + x_{n\cdot k}, p_n)\\ h_{nk}^{(1)}|- &\sim \delta(x_{n\cdot k} = 0)\mathrm{Ber}\left(\frac{\widetilde{\pi}_{nk}}{\widetilde{\pi}_{nk} + (1-\pi_{nk})}\right) + \delta(x_{n\cdot k} > 0),\end{aligned}$$

Authorized licensed use limited to: Rutgers University Libraries. Downloaded on September 15,2024 at 02:31:03 UTC from IEEE Xplore. Restrictions apply.

where $\widetilde{\pi}_{nk} = \pi_{nk}(1-p_n)^{r_k}$, $\pi_{nk} = \sigma(\mathbf{h}_n^{(2)}\mathbf{w}_k^{(1)} + b_k^{(1)})$, $x_{n\cdot k} = \sum_{p=1}^{P} x_{npk}$, $x_{\cdot pk} = \sum_{n=1}^{N} x_{npk}$, and $\zeta_{npk} \propto \phi_{kp}\theta_{nk}$. For the learning of $h_{nk}^{(l)}$ where $l > 1$, the same techniques as in [18] can be used.

Learning Using Stochastic Gradient Thermostats. An alternative way of learning deep PFA is through the use of *stochastic gradient Nóse-Hoover thermostats* (SGNHT), which is more accurate and scalable. Specifically, the following stochastic differential equations (SDE) can be used

$$d\Psi_g = \mathbf{v}dt, d\mathbf{v} = \widetilde{f}(\Psi_g)dt - \xi\mathbf{v}dt + \sqrt{D}d\mathcal{W}$$
$$d\xi = \left(\frac{1}{M}\mathbf{v}^T\mathbf{v} - 1\right)dt,$$

where $\widetilde{f}(\Psi_g) = -\nabla_{\Psi_g}\widetilde{U}(\Psi_g)$ and $\widetilde{U}(\Psi_g)$ is the negative log-posterior of the model. t indexes time and $\mathcal{W}$ denotes the standard Wiener process. ξ is the thermostats variable to make sure the system has a constant temperature. D is the injected variance which is a constant.

4.3.3 Deep Poisson Factor Analysis with Restricted Boltzmann Machine

Similar to the deep PFA above, the restricted Boltzmann machine [24] can be used in place of SBN [17]. If RBM is used, Equations (15) and (16) would be defined using the energy [24]:

$$E(\mathbf{h}_n^{(l)}, \mathbf{h}_n^{(l+1)}) = -\mathbf{h}_n^{(l)}\mathbf{b}_l^T - \mathbf{h}_n^{(l)}\mathbf{W}^{(l)}\mathbf{h}_n^{(l+1)^T} - \mathbf{h}_n^{(l+1)}\mathbf{b}_{l+1}^T.$$

For the learning, similar algorithms as the deep PFA with SBN can be used. Specifically, the sampling process would alternate between $\{\{\phi_k\}, \{\gamma_k\}, \gamma_0\}$ and $\{\{\mathbf{W}_l\}, \{\mathbf{b}_l\}\}$. For $\{\{\phi_k\}, \{\gamma_k\}, \gamma_0\}$, similar conditional density as the SBN-based DPFA is used. For $\{\{\mathbf{W}_l\}, \{\mathbf{b}_l\}\}$, they use the *contrastive divergence* algorithm.

4.3.4 Discussion

In BDL-based topic models, the perception component is responsible for inferring the topic hierarchy from documents while the task-specific component is in charge of modeling the word generation, topic generation, word-topic relation, or inter-document relation. The synergy between these two components comes from the bidirectional interaction between them. On the one hand, knowledge of the topic hierarchy would facilitate accurate modeling of words and topics, providing valuable information for learning inter-document relations. On the other hand, accurately modeling the words, topics, and inter-document relations could help with the discovery of topic hierarchy and learning of compact document representations.

As we can see, the *information exchange* mechanism in some BDL-based topic models is different from that in Section 4.2. For example, in the SBN-based DPFA model, the exchange is natural since the bottom layer of SBN, $\mathbf{H}_1$, and the relationship between $\mathbf{H}_1$ and $\Omega_h = \{\mathbf{X}\}$ are both inherently probabilistic, as shown in Equations (16) and (17), which means additional assumptions about the distribution are not necessary. The SBN-based DPFA model is equivalent to assuming that $\mathbf{H}$ in PFA (see Equation (14)) is generated from a Dirac delta distribution (a Gaussian distribution with zero variance) centered at the bottom layer of the SBN, $\mathbf{H}_1$. Hence both DPFA models in Table 1 are ZV models, according to the definition in Section 4.1. It is worth noting that RSDAE is an HV model (see Equation (11), where $\mathbf{S}$ is the hinge variable and the others are perception variables), and naively modifying this model to be its ZV counterpart would violate the i.i.d. requirement in Section 4.1.

4.4 Other Applications

As mentioned in Section 1, BDL can also be applied to applications beyond data engineering and data mining (e.g., the control of nonlinear dynamical systems from raw images or medical diagnosis with medical images).

Consider controlling a complex dynamical system according to the live video stream received from a camera. One way of solving this control problem is by iteration between two tasks, perception from raw images and control based on dynamic models. The perception task can be taken care of using multiple layers of simple nonlinear transformation (deep learning) while the control task usually needs more sophisticated models such as hidden Markov models and Kalman filters [22], [42]. To enable an effective iterative process between the perception task and the control task, two-way information exchange between them is often necessary. The perception component would be the basis on which the control component estimates its states and on the other hand, the control component with a built-in dynamic model would be able to predict the future trajectory (images) by reversing the perception process. For example, [69] proposed a BDL-based model that performs control based on the received raw images (videos). Their key generative process is as follows:

$$\mathbf{z}_t \sim Q_\phi(Z|X) = \mathcal{N}(\mu_t, \Sigma_t)$$
$$\widetilde{\mathbf{z}}_{t+1} \sim \widetilde{Q}_\psi(\widetilde{Z}|Z, \mathbf{u}) = \mathcal{N}(\mathbf{A}_t\mu_t + \mathbf{B}_t\mathbf{u}_t + \mathbf{o}_t, \mathbf{C}_t)$$
$$\widetilde{\mathbf{x}}_t, \widetilde{\mathbf{x}}_{t+1} \sim P_\theta(X|Z) = Bernoulli(\mathbf{p}_t),$$

where $Q_\phi(Z|X)$ is the encoding model which encodes the raw images X into latent states Z. $\widetilde{Q}_\psi(\widetilde{Z}|Z, \mathbf{u})$ is the transition model which predicts the next latent state $\widetilde{Z}$ given the current latent state Z and the applied control $\mathbf{u}$. $P_\theta(X|Z)$ is the reconstruction (decoding) model which reconstructs the raw images X from latent states Z. The parameters μ_t, Σ_t, $\mathbf{A}_t$, $\mathbf{B}_t$, $\mathbf{o}_t$, $\mathbf{C}_t$, and $\mathbf{p}_t$ are then further parameterized by neural networks.

It is worth noting that in terms of *information exchange* between the two components, this BDL-based control model uses a different mechanism from the ones in Sections 4.2 and 4.3: it uses neural networks to *separately* parameterize the mean and covariance of hinge variables (e.g., μ_t and Σ_t in the encoding model), which is more flexible (with more free parameters) than models such as CDL and CDR in Section 4.2, where Gaussian distributions with fixed variance are also used. Note that this BDL-based control model is an LV model, and since the covariance is assumed to be diagonal [69], the model still meets the i.i.d. requirement in Section 4.1.

5 Conclusion and Future Research

In this paper, we identified a current trend of merging probabilistic graphical models and neural networks (deep

Authorized licensed use limited to: Rutgers University Libraries. Downloaded on September 15,2024 at 02:31:03 UTC from IEEE Xplore. Restrictions apply.

learning), proposed a BDL framework, and reviewed relevant recent work on BDL, which strives to combine the merits of PGM and NN by organically integrating them in a single principled probabilistic framework. To learn parameters in BDL, several algorithms have been proposed, ranging from block coordinate descent, Bayesian conditional density filtering, and stochastic gradient thermostats to stochastic gradient variational Bayes.

BDL has gained its popularity both from the success of PGM and from recent promising advances in deep learning. Since many real-world tasks involve both perception and inference, BDL is a natural choice for harnessing the perception ability from NN and the (causal and logical) inference ability from PGM. Although current applications of BDL focus on recommender systems, topic models, and stochastic optimal control, in the future, we can expect an increasing number of other applications such as link prediction, community detection, active learning, Bayesian reinforcement learning, and many other complex tasks that need interaction between perception and causal inference. In these complex tasks, BDL with interconnected perception components (to handle perception) and task-specific components (to handle inference/reasoning) possesses great performance-boosting potential. Besides, with the advances of efficient *Bayesian neural networks*, BDL with BNN as an important component is expected to be more and more scalable.

References

[1] G. Adomavicius and Y. Kwon, "Improving aggregate recommendation diversity using ranking-based techniques," *IEEE Trans. Knowl. Data Eng.*, vol. 24, no. 5, pp. 896–911, May 2012.

[2] A. K. Balan, V. Rathod, K. P. Murphy, and M. Welling, "Bayesian dark knowledge," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 3420–3428.

[3] I. Bartolini, Z. Zhang, and D. Papadias, "Collaborative filtering with personalized skylines," *IEEE Trans. Knowl. Data Eng.*, vol. 23, no. 2, pp. 190–203, Feb. 2011.

[4] Y. Bengio, L. Yao, G. Alain, and P. Vincent, "Generalized denoising auto-encoders as generative models," in *Proc. Adv. Neural Inf. Process. Syst.*, 2013, pp 899–907.

[5] C. M. Bishop, *Pattern Recognition and Machine Learning*. Secaucus, NJ, USA: Springer, 2006.

[6] D. Blei and J. Lafferty, "Correlated topic models," in *Proc. Adv. Neural Inf. Process. Syst.*, 2006, pp. 147–154.

[7] D. M. Blei and J. D. Lafferty, "Dynamic topic models," in *Proc. 23rd Int. Conf. Mach. Learn.*, 2006, pp 113–120.

[8] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent Dirichlet allocation," *J. Mach. Learn. Res.*, vol. 3, pp. 993–1022, 2003.

[9] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra, "Weight uncertainty in neural network," in *Proc. 32nd Int. Conf. Mach. Learn.*, 2015, pp. 1613–1622.

[10] H. Bourlard and Y. Kamp, "Auto-association by multilayer perceptrons and singular value decomposition," *Biol. Cybern.*, vol. 59, no. 4/5, pp. 291–294, 1988.

[11] Y. Cai, H.-fung Leung, Q. Li, H. Min, J. Tang, and J. Li, "Typicality-based collaborative filtering recommendation," *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 3, pp. 766–779, Mar. 2014.

[12] M. Chen, K. Q. Weinberger, F. Sha, and Y. Bengio, "Marginalized denoising auto-encoders for nonlinear representations," in *Proc. 32nd Int. Conf. Mach. Learn.*, 2014, pp. 1476–1484.

[13] M. Chen, Z. E. Xu, K. Q. Weinberger, and F. Sha, "Marginalized denoising autoencoders for domain adaptation," in *Proc. 29th Int. Conf. Mach. Learn.*, 2012, pp. 767–774.

[14] T. Chen, W. Zhang, Q. Lu, K. Chen, Z. Zheng, and Y. Yu, "SVDFeature: A toolkit for feature-based collaborative filtering, *J. Mach. Learn. Res.*, vol. 13, no. 1, pp. 3619–3622, 2012.

[15] Y. Gal and Z. Ghahramani, "Dropout as a Bayesian approximation: Insights and applications," in *Proc. Int. Conf. Mach. Learn. Deep Learn. Workshop*, 2015.

[16] M. J. F. Gales and S. S. Airey, "Product of Gaussians for speech recognition," *Comput. Speech Language*, vol. 20, no. 1, pp. 22–40, 2006.

[17] Z. Gan, C. Chen, R. Henao, D. E. Carlson, and L. Carin, "Scalable deep Poisson factor analysis for topic modeling," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 1823–1832.

[18] Z. Gan, R. Henao, D. E. Carlson, and L. Carin, "Learning deep sigmoid belief networks with data augmentation," in *Proc. 18th Int. Conf. Artif. Intell. Statistics*, 2015, pp. 267–276.

[19] K. Georgiev and P. Nakov, "A non-IID framework for collaborative filtering with restricted Boltzmann machines," in *Proc. 30th Int. Conf. Mach. Learn.*, 2013, pp. 1148–1156.

[20] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016.

[21] A. K. Gupta and D. K. Nagar, *Matrix Variate Distributions*. London, U.K.: Chapman & Hall, 2000.

[22] J. Harrison and M. West, *Bayesian Forecasting & Dynamic Models*. New York, NY, USA: Springer, 1999.

[23] J. M. Hernández-Lobato and R. Adams, "Probabilistic backpropagation for scalable learning of Bayesian neural networks," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 1861–1869.

[24] G. E. Hinton, "Training products of experts by minimizing contrastive divergence," *Neural Comput.*, vol. 14, no. 8, pp. 1771–1800, 2002.

[25] G. E. Hinton and D. Van Camp, "Keeping the neural networks simple by minimizing the description length of the weights," in *Proc. 6th Annu. Conf. Comput. Learn. Theory*, 1993, pp. 5–13.

[26] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[27] M. Hoffman, F. R. Bach, and D. M. Blei, "Online learning for latent Dirichlet allocation," in *Proc. Adv. Neural Inf. Process. Syst.*, 2010, pp. 856–864.

[28] M. D. Hoffman, D. M. Blei, C. Wang, and J. W. Paisley, "Stochastic variational inference," *J. Mach. Learn. Res.*, vol. 14, no. 1, pp. 1303–1347, 2013.

[29] M. F. Hornick and P. Tamayo, "Extending recommender systems for disjoint user/item sets: The conference recommendation problem," *IEEE Trans. Knowl. Data Eng.*, vol. 24, no. 8, pp. 1478–1490, Aug. 2012.

[30] Y. Hu, Y. Koren, and C. Volinsky, "Collaborative filtering for implicit feedback datasets," in *Proc. 18th IEEE Int. Conf. Data Mining*, 2008, pp. 263–272.

[31] D. H. Hubel and T. N. Wiesel, "Receptive fields and functional architecture of monkey striate cortex," *J. Physiology*, vol. 195, no. 1, pp. 215–243, 1968.

[32] M. I. Jordan, Z. Ghahramani, T. Jaakkola, and L. K. Saul, "An introduction to variational methods for graphical models," *Mach. Learn.*, vol. 37, no. 2, pp. 183–233, 1999.

[33] N. Kalchbrenner, E. Grefenstette, and P. Blunsom, "A convolutional neural network for modelling sentences," in *Proc. 52nd Annu. Meeting Assoc. Comput. Linguistics*, 2014, pp. 655–665.

[34] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *CoRR*, vol. abs/1312.6114, http://arxiv.org/abs/1312.6114, 2013.

[35] Y. LeCun, "Modeles connexionnistes de l'apprentissage (connectionist learning models)," PhD dissertation, Dept. of Computer Science, Université P. et M. Curie, Paris 6, France, Jun. 1987.

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

[37] S. Li, J. Kawale, and Y. Fu, "Deep collaborative filtering via marginalized denoising auto-encoder," in *Proc. 24th ACM Int. Conf. Inf. Knowl. Manage.*, 2015, pp. 811–820.

[38] W.-J. Li and D.-Y. Yeung, "Relation regularized matrix factorization," in *Proc. 21st Int. Joint Conf. Artif. Intell.*, 2009, pp. 1126–1131.

[39] N. N. Liu, X. Meng, C. Liu, and Q. Yang, "Wisdom of the better few: Cold start recommendation via representative based rating elicitation, in *Proc. 5th ACM Conf. Recommender Syst.*, 2011, pp. 37–44.

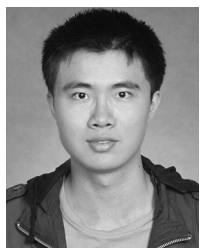
[40] Z. Lu, Z. Dou, J. Lian, X. Xie, and Q. Yang, "Content-based collaborative filtering for news topic recommendation," in *Proc. 29th AAAI Conf. Artif. Intell.*, 2015, pp. 217–223.

[41] J. C. MacKay David, "A practical Bayesian framework for backprop networks," *Neural Comput.*, vol. 4, no. 3, pp. 448–472, 1992.

[42] T. Matsubara, V. Gómez, and H. J. Kappen, "Latent Kullback Leibler control for continuous-state systems using probabilistic graphical models," in *Proc. 30th Conf. Uncertainty Artif. Intell.*, 2014, pp. 583–592.

Authorized licensed use limited to: Rutgers University Libraries. Downloaded on September 15,2024 at 02:31:03 UTC from IEEE Xplore. Restrictions apply.

[43] R. M. Neal, "Connectionist learning of belief networks," *Artif. Intell.*, vol. 56, no. 1, pp. 71–113, 1992.
[44] R. M. Neal, "Bayesian learning for neural networks," Ph.D. dissertation, Dept. Statistics Comput. Sci., Univ. Toronto, Ontario, Canada, 1995.
[45] A. Van Den Oord, S. Dieleman, and B. Schrauwen, "Deep content-based music recommendation," in *Proc. Adv. Neural Inf. Process. Syst.*, 2013, pp. 2643–2651.
[46] Y.-J. Park, "The adaptive clustering method for the long tail problem of recommender systems," *IEEE Trans. Knowl. Data Eng.*, vol. 25, no. 8, pp. 1904–1915, Aug. 2013.
[47] I. Porteous, D. Newman, A. Ihler, A. Asuncion, P. Smyth, and M. Welling, "Fast collapsed Gibbs sampling for latent Dirichlet allocation," in *Proc. 14th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2008, pp. 569–577.
[48] C. Poultney, S. Chopra, and Y. L. Cun, "Efficient learning of sparse representations with an energy-based model," in *Proc. Adv. Neural Inf. Process. Syst.*, 2006, pp. 1137–1144.
[49] S. Purushotham, Y. Liu, and C.-C. Jay Kuo, "Collaborative topic regression with social matrix factorization for recommendation systems," in *Proc. 29th Int. Conf. Mach. Learn.*, 2012, pp. 759–766.
[50] F. Ricci, L. Rokach, and B. Shapira, *Introduction to Recommender Systems Handbook*. Berlin, Germany: Springer, 2011.
[51] S. Rifai, P. Vincent, X. Muller, X. Glorot, and Y. Bengio, "Contractive auto-encoders: Explicit invariance during feature extraction," in *Proc. 28th Int. Conf. Mach. Learn.*, 2011, pp. 833–840.
[52] T. N. Sainath, B. Kingsbury, V. Sindhwani, E. Arisoy, and B. Ramabhadran, "Low-rank matrix factorization for deep neural network training with high-dimensional output targets," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, 2013, pp. 6655–6659.
[53] R. Salakhutdinov and A. Mnih, "Probabilistic matrix factorization," in *Proc. Adv. Neural Inf. Process. Syst.*, 2007, pp. 1257–1264.
[54] R. Salakhutdinov, A. Mnih, and G. E. Hinton, "Restricted Boltzmann machines for collaborative filtering," in *Proc. 24th Int. Conf. Mach. Learn.*, 2007, pp. 791–798.
[55] J. R. Shewchuk, "An introduction to the conjugate gradient method without the agonizing pain," Carnegie Mellon Univ., Pittsburgh, PA, USA, Tech. Rep. CMU-CS-94–125, 1994.
[56] A. P. Singh and G. J. Gordon, "Relational learning via collective matrix factorization," in *Proc. 14th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2008, pp. 650–658.
[57] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, 2014.
[58] R. S. Strichartz, *A Guide to Distribution Theory and Fourier Transforms*. Singapore: World Scientific, 2003.
[59] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," *J. Mach. Learn. Res.*, vol. 11, pp. 3371–3408, 2010.
[60] C. Wang and D. M. Blei, "Collaborative topic modeling for recommending scientific articles," in *Proc. 17th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2011, pp. 448–456.
[61] C. Wang, D. M. Blei, and D. Heckerman, "Continuous time dynamic topic models," in *Proc. 24th Conf. Uncertainty Artif. Intell.*, 2008, pp. 579–586.
[62] H. Wang, B. Chen, and W.-J. Li, "Collaborative topic regression with social regularization for tag recommendation," in *Proc. 23rd Int. Joint Conf. Artif. Intell.*, 2013, pp. 2719–2725.
[63] H. Wang and W.-J. Li, "Relational collaborative topic regression for recommender systems," *IEEE Trans. Knowl. Data Eng.*, vol. 27, no. 5, pp. 1343–1355, May 2015.
[64] H. Wang, X. Shi, and D.-Y. Yeung, "Relational stacked denoising autoencoder for tag recommendation," in *Proc. 29th AAAI Conf. Artif. Intell.*, 2015, pp. 3052–3058.
[65] H. Wang, X. Shi, and D.-Y. Yeung, "Collaborative recurrent autoencoder: Recommend while learning to fill in the blanks," in *Proc. 13th Annu. Conf. Neural Inf. Process. Syst.*, 2016, to be published.
[66] H. Wang, X. Shi, and D.-Y. Yeung, "Natural parameter networks: A class of probabilistic neural networks," in *Proc. 13th Annu. Conf. Neural Inf. Process. Syst.*, 2016, to be published.
[67] H. Wang, N. Wang, and D.-Y. Yeung, "Collaborative deep learning for recommender systems," in *Proc. 21th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2015, pp. 1235–1244.
[68] X. Wang and Y. Wang, "Improving content-based and hybrid music recommendation using deep learning," in *Proc. 22nd ACM Int. Conf. Multimedia*, 2014, pp. 627–636.
[69] M. Watter, J. Springenberg, J. Boedecker, and M. Riedmiller, "Embed to control: A locally linear latent dynamics model for control from raw images," in *Proc. 28th Int. Conf. Neural Inf. Process. Syst.*, 2015, pp. 2728–2736.
[70] Y. Z. Wei, L. Moreau, and N. R. Jennings, "Learning users' interests by quality classification in market-based recommender systems," *IEEE Trans. Knowl. Data Eng.*, vol. 17, no. 12, pp. 1678–1688, Dec. 2005.
[71] H. Ying, L. Chen, Y. Xiong, and J. Wu, "Collaborative deep ranking: A hybrid pair-wise recommendation algorithm with implicit feedback," in *Proc. 20th Pacific-Asia Conf. Adv. Knowl. Discovery Data Mining*, 2016, pp. 555–567.
[72] F. Zhang, N. J. Yuan, D. Lian, X. Xie, and W.-Y. Ma, "Collaborative knowledge base embedding for recommender systems," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2016, pp. 353–362.
[73] V. W. Zheng, B. Cao, Y. Zheng, X. Xie, and Q. Yang, "Collaborative filtering meets mobile recommendation: A user-centered approach," in *Proc. 24th AAAI Conf. Artif. Intell.*, 2010, pp. 236–242.
[74] M. Zhou and L. Carin, "Negative binomial process count and mixture modeling," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 2, pp. 307–320, Feb. 2015.
[75] M. Zhou, L. Hannah, D. B. Dunson, and L. Carin, "Beta-negative binomial process and Poisson factor analysis," in *Proc. 15th Int. Conf. Artif. Intell. Statistics*, 2012, pp. 1462–1471.



Hao Wang received the BSc degree in computer science from Shanghai Jiao Tong University, China. He is currently working toward the PhD degree in the Department of Computer Science and Engineering, Hong Kong University of Science and Technology. His research interests include statistical machine learning and data mining. He received the Hong Kong PhD Fellowship in 2013, the Microsoft Research Asia Fellowship in 2015, and the Baidu Research Fellowship in 2015 for his achievements on statistical machine learning and Bayesian deep learning.

Dit-Yan Yeung received the BEng degree in electrical engineering and MPhil degree in computer science from the University of Hong Kong, and the PhD degree in computer science from the University of Southern California. He started his academic career as an assistant professor with the Illinois Institute of Technology, Chicago. He then joined the Hong Kong University of Science and Technology where he is now a full professor with the Department of Computer Science and Engineering, with joint appointment in the Department of Electronic and Computer Engineering. His research interests include computational and statistical approaches to machine learning and artificial intelligence. He is a senior member of the IEEE.

▷ **For more information on this or any other computing topic, please visit our Digital Library at** www.computer.org/publications/dlib.

Authorized licensed use limited to: Rutgers University Libraries. Downloaded on September 15,2024 at 02:31:03 UTC from IEEE Xplore. Restrictions apply.