Hyper-Parameter-Free Generative Modelling with Deep Boltzmann Trees

Nico Piatkowski (✉)

TU Dortmund, AI Group
nico.piatkowski@tu-dortmund.de

Abstract. Deep neural networks achieve state-of-the-art results in various classification and synthetic data generation tasks. However, only little is known about why depth improves a model. We investigate the structure of stochastic deep neural works, also known as Deep Boltzmann Machines, to shed some light on this issue. While the best known results postulate an exponential dependence between the number of visible units and the depth of the model, we show that the required depth is upper bounded by the longest path in the underlying junction tree, which is at most *linear* in the number of visible units. Moreover, we show that the conditional independence structure of any categorical Deep Boltzmann Machine contains a sub-tree that allows the consistent estimation of the full joint probability mass function of all visible units. We connect our results to l_1-regularized maximum-likelihood estimation and Chow-Liu trees. Based on our theoretical findings, we present a new tractable version of Deep Boltzmann Machines, namely the *Deep Boltzmann Tree (DBT)*. We provide a hyper-parameter-free algorithm for learning the DBT from data, and propose a new initialization method to enforce convergence to good solutions. Our findings provide some theoretical evidence for why a deep model might be beneficial. Experimental results on benchmark data show, that the DBT is a theoretical sound alternative to likelihood-free generative models.

Keywords: Deep Boltzmann Machine · Structure Learning · Generative Model

1 Introduction

Modern applications of data science necessitate expressive, robust and efficient probabilistic models, to capture the rich structure in complex data sets. These models generally fall into two major categories: likelihood-based and likelihood-free. The former explicitly assigns a likelihood function $\mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{X})$ with parameters $\boldsymbol{\theta}$ to describe the data $\boldsymbol{X}$, while the latter learns a model from which samples from the desired distribution may be drawn (but does not assign or learn a form for the distribution itself). In this work, we study deep generative models w.r.t. their structure, also known as network architecture.

Specifically, likelihood-free methods typically pass samples $\boldsymbol{z}$ from a pre-specified simple distribution $q(\boldsymbol{z})$ through a deterministic mapping $G(\boldsymbol{z};\boldsymbol{\theta})$:

$\mathcal{Z} \to \mathcal{X}$, commonly known as the generator. While likelihood-free methods gain a lot of attention, they lack theoretical insights on almost every fundamental aspect, including model selection, parameter learning, and sample complexity. Selecting the right model suffers from an countable infinite search space. In most cases, training such generative adversarial networks [10] is cumbersome and involves sophisticated hyper-parameter tuning strategies. However, generalization bounds [18, 2, 9] which quantify the model's error w.r.t. inherent properties, like depth and width of the underlying neural network, can guide this process. Other research directions try to bring likelihood and entropy back in implicitly defined generative models [25, 13].

In contrast, likelihood-based methods enjoy theoretical insights and statistical guarantees, but suffer from a high computational complexity. To bridge the gap between popular deep generative models and classic probabilistic models, we consider Deep Boltzmann Machines (DBMs) [20] with arbitrary categorical hidden state spaces, as a generic class of stochastic neural networks. They have their roots in statistical physics and have been studied intensively as special types of graphical model. In particular, information geometry has provided deep geometric insights about learning and approximation of probability distributions by this kind of networks. It is well known that general Boltzmann machines are universal approximators of probability distributions over the states of their visible units, provided they have sufficiently many hidden units. Moreover, the universal approximation capability has been shown for Restricted Boltzmann Machines, provided their single hidden layer has exponentially more units than the visible layer. In a similar way, universal approximation results for DBMs suggest that the number of layers should be exponential in the number of visible units [16].

However, practical deep models are far from having exponentially many layers, still providing superior quality. Driven by this apparent contradiction, we study the structure of DBMs to gain new theoretical insights about deep probabilistic models in general. Our findings guide us to a new model class: the Deep Boltzmann Tree (DBT). Like a DBM, a DBT has one layer of visible (input) units and multiple hidden layers, containing latent variables (Fig. 1). Unlike a DBM, the structure of a DBT contains no loops, and thus, allows for tractable, poly-time probabilistic inference.

Our contributions can be summarized as follows:

- We state a new universal approximation theorem for Deep Boltzmann Machines, which shows that the dependence between the number of layers and the number of visible units is at most linear.
- We define a new, tractable type of deep probabilistic model: the Deep Boltzmann Tree. We show that DBTs are universal approximators and connect them to results in structure learning.
- We provide hyper-parameter-free algorithms for constructing and learning DBTs. Here, hyper-parameter-free means that "magic constants" like learning rate, model architecture, and hidden state space, are automatically determined from data. The proposed method has literally no tuneable parameter.

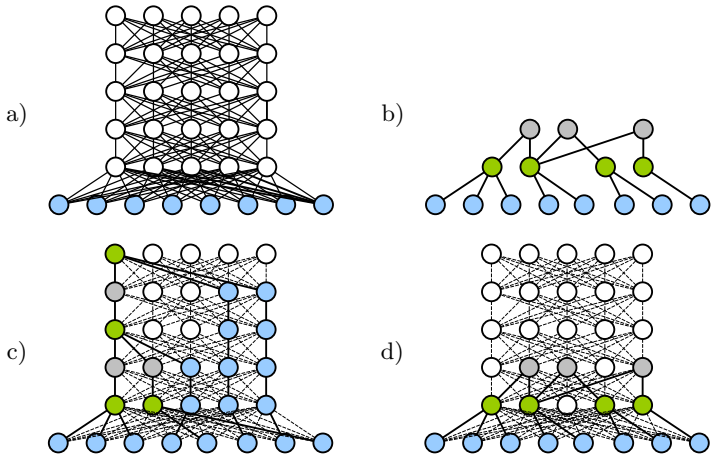


Fig. 1. The conditional independence structure of a Deep Boltzmann Machine with $n_0 = 8$ visible units and 5×5 hidden units is shown in a). Dashed edges and white neurons in c) and d) are not required to represent the full joint probability mass function—they can be dropped by nullifying the corresponding edge weights $\boldsymbol{\theta}_e \leftarrow \mathbf{0}$. The colored neuron in c) and d) both correspond to the DBT shown in b). Blue hidden units copy the state of one visible unit to a deeper layer by setting the corresponding edge weight $\boldsymbol{\theta}_e$ to the identity function $\boldsymbol{\theta}_{\mathrm{id}}$.

2 Notation and Background

Let us summarize the notation and background necessary for the subsequent development. The Kullback-Leibler divergence between two probability mass functions $\mathbb{P}$ and $\mathbb{Q}$ is defined by $\mathrm{KL}[\mathbb{Q}\|\mathbb{P}] = \sum_{\boldsymbol{x}\in\mathcal{X}} \mathbb{Q}(\boldsymbol{x})(\log \mathbb{Q}(\boldsymbol{x}) - \log \mathbb{P}(\boldsymbol{x}))$, which is never negative and only zero if and only if $\mathbb{P} = \mathbb{Q}$. If f is a function, f^{-1} refers to its inverse.

2.1 Graphical Models

An undirected graph $G = (V, E)$ consists of $n = |V|$ vertices, connected via edges $(v, w) \in E$. For two graphs G_1, G_2, we write $V(G_1)$ and $V(G_2)$ to denote the vertices of G_1 and G_2, respectively and similar $E(G_1)$ and $E(G_2)$ for the edges. A clique C is a fully-connected subset of vertices, i.e., $\forall v, w \in C : (v, w) \in E$. The set of all cliques of G is denoted by $\mathcal{C}$. Here, any undirected graph represents the conditional independence structure of an undirected graphical model or Markov random field [24], shown in Fig. 2 a). To this end, we identify each vertex $v \in V$ with a random variable $\boldsymbol{X}_v$ taking values in the state space $\mathcal{X}_v$. The random vector $\boldsymbol{X} = (\boldsymbol{X}_v : v \in V)$, with probability mass function (pmf) $\mathbb{P}$, represents the random joint state of all vertices in some arbitrary but fixed order, taking values $\boldsymbol{x}$ in the Cartesian product space $\mathcal{X} = \bigotimes_{v\in V} \mathcal{X}_v$. If not stated otherwise, $\mathcal{X}$ is a discrete set. Moreover, we allow to access these quantities for any proper

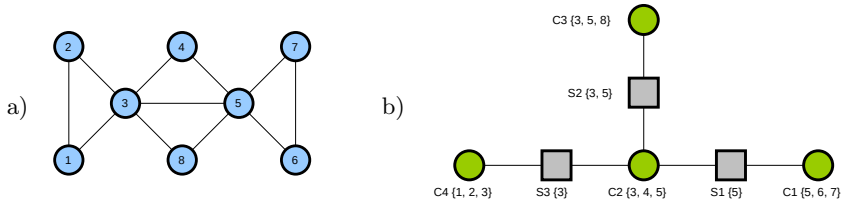


Fig. 2. The conditional independence structure a) of the underlying random variable $\boldsymbol{X}$ is first converted to the junction tree b).

subset of variables $S \subset V$, i.e., $\boldsymbol{X}_S = (\boldsymbol{X}_v : v \in S)$, $\boldsymbol{x}_S$, and $\mathcal{X}_S$, respectively. We write $C_{\max}$ for the clique C that has the largest state space $\mathcal{X}_C$. According to the Hammersley-Clifford theorem [11], the probability mass of $\mathcal{X}$ factorizes over positive functions $\psi_C : \mathcal{X} \to \mathbb{R}_+$, one for each maximal clique of the underlying graph,

$$\mathbb{P}(\boldsymbol{X} = \boldsymbol{x}) = \frac{1}{Z} \prod_{C \in \mathcal{C}} \psi_C(\boldsymbol{x}_C) \,, \tag{1}$$

normalized via $Z = \sum_{\boldsymbol{x} \in \mathcal{X}} \prod_{C \in \mathcal{C}} \psi_C(\boldsymbol{x}_C)$. Due to positivity of ψ_C, it can be written as an exponential, i.e., $\psi_C(\boldsymbol{x}_C) = \exp(\langle \boldsymbol{\theta}_C, \phi_C(\boldsymbol{x}_C) \rangle)$ with sufficient statistic $\phi_C : \mathcal{X}_C \to \mathbb{R}^{|\mathcal{X}_C|}$. The overcomplete sufficient statistic of discrete data is a "one-hot" vector that selects a specific weight value, e.g., $\psi_C(\boldsymbol{x}_C) = \exp(\boldsymbol{\theta}_{C=\boldsymbol{x}_C})$. The full joint can be written in the famous exponential family form $\mathbb{P}(\boldsymbol{X} = \boldsymbol{x}) = \exp(\langle \boldsymbol{\theta}, \phi(\boldsymbol{x}) \rangle - \log Z)$ with $\boldsymbol{\theta} = (\boldsymbol{\theta}_C : C \in \mathcal{C})$ and $\phi(\boldsymbol{x}) = (\phi_C(\boldsymbol{x}_C) : C \in \mathcal{C})$.

The parameters of exponential family members are estimated by minimizing the negative average log-likelihood $\ell(\boldsymbol{\theta}; \mathcal{D}) = -(1/|\mathcal{D}|) \sum_{\boldsymbol{x} \in \mathcal{D}} \log \mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{x})$ for some data set $\mathcal{D}$ via first-order numeric optimization methods. $\mathcal{D}$ contains samples from $\boldsymbol{X}$, and it can be shown that the estimated probability mass converges to the data generating distribution as the size of $\mathcal{D}$ increases. However, computing Z and hence performing probabilistic inference is #P-hard [23, 4]. Exact inference can be carried out via the junction tree algorithm. The junction tree representation of an undirected model is a tree, in which each vertex represents a maximal clique of a chordal completion of G ([24], Sec. 2.5.2). The cutset of each pair of adjacent clique-vertices is called separator. Here, we consider junction trees which contain separators as explicit vertices, as shown in Fig. 2 b).

2.2 Deep Boltzmann Machines

Deep Boltzmann Machines are undirected graphical models for the joint probability mass of an "ordinary" random variable $\boldsymbol{X}$ and a latent variable $\boldsymbol{H}$ that represents a set of so-called *hidden units*. *Latent* means that the value of $\boldsymbol{H}$ cannot be observed and is not contained in the data set $\mathcal{D}$.

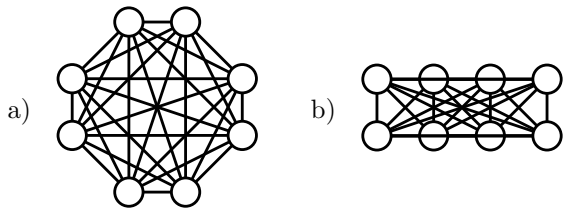


Fig. 3. Two different planar embeddings of the same 8-clique of binary hidden units. Both are equivalent to a single discrete hidden unit with $q = 2^8$ states.

To estimate the parameters in the presence of latent variables, expectation-maximization [6] or contrastive divergence [20] techniques must be applied. In contrast to other undirected models, the conditional independence structure of DBMs is not learned from data. Instead, the connectivity between visible and hidden units as well as between hidden and hidden units is pre-specified and follows the multipartite layered structure that is well known from artificial neural feed-forward networks. An exemplary DBM is shown in Fig. 1 a).

In most cases, the hidden units are assumed to have a binary state space. This is, however, not necessary. We like to stress the fact that DBMs are plain undirected models and as such, any vertex can have any state space. For now, we consider so-called *categorical DBMs*, where all hidden units have the same state space of size q. For convenience, such DBMs are called *q-state DBMs*. A vertex with q states is called *q-state unit*.

Fixing the depth L, the width of each layer $W = (n_1, n_2, \ldots, n_L)$, and the state space size q defines a family of probability mass functions $\mathcal{M}_{L,W,H}$. To measure the expressive power of such a family, we resort to the same notion of approximation guarantee that is used in the DBM literature, e.g., [16].

Definition 1 (Universal Approximation). *A set $\mathcal{M}$ of probability mass functions on $\mathcal{X}$ is called universal approximator when, for any probability mass $\mathbb{Q}$ on $\mathcal{X}$ and any $\epsilon > 0$, there is a pmf $\mathbb{P}$ in $\mathcal{M}$ with* $\mathrm{KL}[\mathbb{Q}\|\mathbb{P}] \leq \epsilon$.

An obvious question is which choices of L, W, and q make a DBM an universal approximator. A rather indirect way to explain this is the identification of settings in which the (undirected) DBM can be treated as if it is a directed (feed-forward) network [14, 15]. While the required proof technique is rather cumbersome and mathematically involved, this point of view paves the way to the best currently known result on the depth of DBMs:

Theorem 1 (DBMs are Universal Approximators with Exponential Dependence on n_0 [15]). *Let M be a q-state DBM with n_0 q-state visible units and L hidden layers of n_0 units each. Let further $\boldsymbol{X} = (\boldsymbol{X}_1, \boldsymbol{X}_2, \ldots, \boldsymbol{X}_{n_0})$ be the random variable that represents all visible units. Then, M is a universal approximator for $\mathbb{P}(\boldsymbol{X})$ provided L is large enough. More precisely, for any $n_0 \leq n := q^k + k + 1$, for some $k \in \mathbb{N}$, a sufficient condition is $L \geq 1 + (q^n + 1)/(q(q-1)(n - \log_q(n) - 1))$. For any n_0, a necessary condition is $L \geq (q^{n_0} - 1)/(n_0(q-1)(n_0(q-1)+2))$.*

A closer look at the necessary condition suggests that this result is rather pessimistic: for $n_0 = 16$ and $q = 2$, we have $L \geq 227$—a fairly deep model for 16 binary inputs. Considering an MNIST-scale binary input, i.e., $n_0 = 784$, we have $L \geq 1.6511 \times 10^{230}$—an astronomically deep network when compared to state-of-the-art architectures [12, 22].

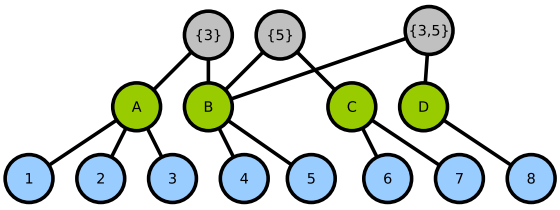


Fig. 4. An exemplary Deep Boltzmann Tree with 8 visible units (blue), 4 hidden clique-units (green), and 3 hidden separator-units (grey). The separator-hidden units are annotated with the separating vertices, i.e., with the intersection of their incident cliques in the underlying junction tree.

A disturbing fact about the above theorem is that a larger latent state space, i.e., increasing q, does not decrease the required depth of the network. Instead, the theorem tells us that a *deeper* network is required. This is especially odd because a single hidden unit with $q = b^k$ states can be reinterpreted as a fully connected set of k hidden units having b states each. As shown in Fig. 3, we can rearrange the clique to emulate 2 DBM layers with inter-layer connections. Thus, increasing the state space of hidden units is *equivalent to increasing the depth*! Hence, a meaningful lower bound on the depth of a network should *decrease* with the expressivity of the hidden units. Driven by this observation, we exploit classic insights about conditional independence structures to derive a new model class as well as new theoretical insights on the depth of q-state DBMs.

3 Deep Boltzmann Trees

Deep learning architectures are ubiquitous, mostly application specific, and validated on benchmark data. Theoretical justifications are usually replaced by superior benchmark results. Stochastic DBM-based architectures inherit their computational complexity from ordinary graphical models which renders exact inference intractable and forces the user to resort to Markov chain Monte Carlo techniques [21].

In contrast, we present a generic deep architecture that can be learned from data. In what follows, we explain the learning procedure and prove that the learned model can approximate the true underlying probability mass function with arbitrary small error. As a by-product, we obtain the best known bounds on the depth required by any q-state DBM to be an universal approximator.

Our proposed model class is called Deep[1] Boltzmann Tree. The algorithmic procedure for the construction of DBTs is provided in Alg. 1. An exemplary DBT

[1] It turns out that DBTs consist of exactly two hidden layers. While this kind of depth seems rather "shallow", original work on DBMs [20] define the DBM as a restricted

Algorithm 1: Constructing the Deep Boltzmann Tree

Require: Conditional independence structure G
1: $J \leftarrow$ Junction tree of G
2: $V \leftarrow V(J)$
3: $E \leftarrow E(J)$
4: **for** clique vertices $C \in J$ **do**
5: **for** vertices $v \in C$ **do**
6: **if** $v \notin V$ **then**
7: $E \leftarrow E \cup \{(v, C)\}$
8: $V \leftarrow V \cup \{v\}$
9: **end if**
10: **end for**
11: **end for**
12: **return** $T = (V, E)$ // The DBT

is shown in Fig. 4. While the DBT architecture relies heavily on the junction tree structure, it is important to understand that all vertices inherited from J (line 2 of Alg. 1) represent hidden units (latent variables) in the DBT. This difference is of utmost importance: plain junction tree models enforce clique states which do not appear in the training data to be unlikely. Instead, DBTs are capable of learning that the probability mass of unknown states is similar to that of some known states.

Moreover, we make no use of specialized junction tree inference algorithms, like Shafer-Shenoy algorithm or Hugin algorithm [24]. The asymptotic runtime of Alg. 1 is TIME(JT)+$\mathcal{O}(n|C_{\max}|)$, where TIME(JT) is the runtime of the junction tree construction and $|C_{\max}|$ is 1 plus the tree-width of the input graph. Here, an (NP-complete) minimum chordal completion is not required—any non-minimal poly-time triangulation suffices.

The joint pmf of visible and hidden units can then be written as

$$\mathbb{P}_T(\boldsymbol{X} = \boldsymbol{x}, \boldsymbol{H} = \boldsymbol{h}) = \frac{1}{Z_T} \prod_{(u,C)\in E_{\mathcal{UC}}} \psi_{(u,C)}(\boldsymbol{x}_u, \boldsymbol{h}_C) \prod_{(C,S)\in E_{\mathcal{CS}}} \psi_{(C,S)}(\boldsymbol{h}_C, \boldsymbol{h}_S) \;, \tag{2}$$

where $E_{\mathcal{UC}} = E(T) \cap \mathcal{U} \times \mathcal{C}$, $E_{\mathcal{CS}} = E(T) \cap \mathcal{C} \times \mathcal{S}$, $V(T) = \mathcal{U} \cup \mathcal{C} \cup \mathcal{S}$, $\mathcal{U}$ being the set of visible units, $\mathcal{C}$ being the set of hidden clique-units, and $\mathcal{S}$ being the set of hidden separator-units—visualized in Fig. 4 by blue, green, and grey vertices, respectively. The random variable $\boldsymbol{H}$ represents the random joint realization of all hidden (green and grey) units. Note that Eq. 2 arises from the general factorization of undirected graphical models Eq. 1, having maximal cliques of size two only.

Being an undirected graphical model, the DBT pmf can be written in exponential family form, and as such, it obeys a canonical parametrization in terms

Boltzmann machine which has *more than one* hidden layer. Thus, to be consistent with the common terminology, we decided to denote our proposed model as "deep".

of edge weights. We will now exploit this parametrization to declare universal approximation for any DBT with sufficiently large hidden state space.

Theorem 2 (Deep Boltzmann Trees are Universal Approximators). *Let $\mathbb{P}_G$ be the full joint pmf of the random variable $\boldsymbol{X}$ with conditional independence structure G. Let further T be the output of Alg. 1. When the state space of each DBT hidden unit is at least $|\mathcal{X}_{C_{\max}}|$, then there exists a canonical weight vector $\boldsymbol{\theta}$, such that*

$$\mathrm{KL}[\mathbb{P}_G \| \mathbb{P}'_T] \leq \epsilon$$

for any $\epsilon > 0$ and with $\mathbb{P}'_T(\boldsymbol{x}) = \sum_{\boldsymbol{h}} \mathbb{P}_T(\boldsymbol{x}, \boldsymbol{h})$.

Proof. Let $\boldsymbol{\theta}_{\mathcal{UC}} = (\boldsymbol{\theta}_{(v,C)} : E_{\mathcal{UC}})$ with $E_{\mathcal{UC}} = E(T) \cap \mathcal{U} \times \mathcal{C}$ contain all DBT weight vectors for edges that connect a visible vertex with a clique vertex. In a similar way, let $\boldsymbol{\theta}_{\mathcal{CS}} = (\boldsymbol{\theta}_{(C,S)} : E_{\mathcal{CS}})$ with $E_{\mathcal{CS}} = E(T) \cap \mathcal{C} \times \mathcal{S}$ contain all DBT weight vectors for edges that connect a clique vertex with a separator vertex. Finally, let $\boldsymbol{\theta}^* = (\boldsymbol{\theta}_C : C \text{ is maximal clique in chordal completion of } G)$ denote the clique weights of a chordal completion of G. In other words, $\boldsymbol{\theta}^*$ contains the junction tree weights. We will now choose values for $\boldsymbol{\theta}_{\mathcal{UC}}$ and $\boldsymbol{\theta}_{\mathcal{CS}}$ which guarantee the conclusion of the theorem.

Any hidden unit of T corresponds to a clique or separator vertex of the junction tree. Each hidden clique-unit $F \in V(T)$ is connected to visible units and hidden separator-units only. We do now abuse notation and identify F with the union of its visible neighbors and the content of their neighboring hidden separator-units. E.g., if $F = D$ in Fig. 4, we have $F = \{3, 5, 8\}$.

Let us fix some constant $\omega \in \mathbb{R}_+$. We will now assign two types of edge weights to any hidden clique-unit F:

(I) Each hidden clique-unit is incident to *exactly one edge* of type I—it is irrelevant which of the incident edges. Type I edges simulate the original junction tree weight $\boldsymbol{\theta}^*_F$ of the junction tree vertex F. The precondition of the theorem guarantees that the state space of the DBT unit F is at least as large as the state space of the corresponding clique in the chordal completion of G. Thus, there exist an injective function ρ, that maps the joint state of F's neighbors to exactly one of F's states. Assume that v is a neighbor of F and consider the edge (v, F). When (v, F) is a type I edge, then, for each weight $\boldsymbol{\theta}_{(v=x,F=y)}$ we have $\boldsymbol{\theta}_{(v=x,F=y)} = \boldsymbol{\theta}^*_{F=y}$ if and only if $\rho^{-1}(y)_v = x$, e.g., the joint state of F's neighbors that corresponds to y agrees with $v = x$. Otherwise, we have $\boldsymbol{\theta}_{(v=x,F=y)} = -\omega$. Moreover, for all weights $\boldsymbol{\theta}_{(v=x,F=y')}$ which correspond to hidden states y' that have no corresponding joint state, i.e., when the hidden state space is larger than the number of clique states, we set $\boldsymbol{\theta}_{(v=x,F=y')} = -\omega$.

(II) When (v, F) is a type II edge, then, for each weight $\boldsymbol{\theta}_{(v=x,F=y)}$ we have $\boldsymbol{\theta}_{(v=x,F=y)} = 0$ if and only if $\rho^{-1}(y)_v = x$. Otherwise, we have $\boldsymbol{\theta}_{(v=x,F=y)} = -\omega$. Again, we set all weights $\boldsymbol{\theta}_{(v=x,F=y')}$ which correspond to hidden states y' that have no corresponding joint state to $-\omega$.

For both edge types, we call edge states whose weight is $-\omega$ *not realizable*, and otherwise *realizable*. The concept of realizable edge states extends naturally to full joint states, i.e., whenever a joint state $(\boldsymbol{x}, \boldsymbol{h})$ of visible and hidden units contains at least one not realizable edge state, $(\boldsymbol{x}, \boldsymbol{h})$ itself is not realizable. Let $\mathcal{R}$ denote the set of all realizable joint states and $\overline{\mathcal{R}}$ its complement. Having that said, let us investigate the partition function Z_T of (Eq. 2):

$$Z_T = \sum_{(\boldsymbol{x},\boldsymbol{h})} \prod_{(u,C)\in E_{\mathcal{U}\mathcal{C}}} \psi_{(u,C)}(\boldsymbol{x}_u, \boldsymbol{h}_C) \prod_{(C,S)\in E_{\mathcal{C}\mathcal{S}}} \psi_{(C,S)}(\boldsymbol{h}_C, \boldsymbol{h}_S) \ .$$

Now, partition the summation w.r.t. realizability, and observe that each factor of a realizable joint state is either $\exp(0) = 1$ or $\exp(\boldsymbol{\theta}^*_{F=\rho(\boldsymbol{x}_F)})$:

$$\begin{aligned} Z_T = &\sum_{\boldsymbol{x}} \prod_{F} \exp(\boldsymbol{\theta}^*_{F=\rho(\boldsymbol{x}_F)}) + \\ &\sum_{(\boldsymbol{x},\boldsymbol{h})\in\overline{\mathcal{R}}} \prod_{(u,C)\in E_{\mathcal{U}\mathcal{C}}} \psi_{(u,C)}(\boldsymbol{x}_u, \boldsymbol{h}_C) \prod_{(C,S)\in E_{\mathcal{C}\mathcal{S}}} \psi_{(C,S)}(\boldsymbol{h}_C, \boldsymbol{h}_S) \ . \end{aligned}$$

In the limit of $\omega \to \infty$, the sum over not realizable states vanishes (because $exp(-\omega) \to 0$) and Z_T converges to the partition function of the ordinary junction tree factorization. In the same way, $\lim_{\omega\to\infty} \mathbb{P}_T(\boldsymbol{x}, \boldsymbol{h})$ converges either to 0 whenever $(\boldsymbol{x}, \boldsymbol{h}) \in \overline{\mathcal{R}}$, or to $\mathbb{P}_J(\boldsymbol{x})$ whenever $(\boldsymbol{x}, \boldsymbol{h}) \in \mathcal{R}$. Since the junction tree pmf $\mathbb{P}_J$ is identical to the original undirected model $\mathbb{P}_G$, we have $\lim_{\omega\to\infty} \sum_{\boldsymbol{h}} \mathbb{P}_T(\boldsymbol{x}, \boldsymbol{h}) = \mathbb{P}_G(\boldsymbol{x})$. Thus, for any $\epsilon > 0$, there exists $\omega > 0$ such that $\mathrm{KL}[\mathbb{P}_G \| \mathbb{P}'_T] \leq \epsilon$. □

The theorem tells us that it is always possible to find DBT weights $\boldsymbol{\theta}$ which make the approximation error arbitrarily small as long as the DBT's latent state space is large enough. Surprisingly, the result carries over to q-state DBMs. The idea is to embed the DBT into the DBM as visualized in Fig. 1 c) and d).

Theorem 3 (DBMs are Universal Approximators with Linear Dependence on n_0). *Let M be a q-state DBM with n_0 visible units and L hidden layers of n_0 units each. Let further $\boldsymbol{X}$ be the random variable that represents all visible units. Then, M is a universal approximator for $\mathbb{P}(\boldsymbol{X})$ provided L and q are large enough. More precisely, if $q \geq |\mathcal{X}_{C_{\max}}|$, it suffices that $L \geq 2$.*

Proof. Notice that the output T of Alg. 1 is an especially simple tripartite graph, indicated by the coloring in Fig. 4. By identifying the visible units of T with the visible units of M, the remainder is a bipartite graph that consists of hidden clique-units and hidden separator-units. The precondition of the theorem asserts that each hidden layer has n_0 units. Each hidden clique-unit of T arises from some maximal clique of a chordal completion of the true conditional independence structure of $\boldsymbol{X}$. The number of maximal cliques in a chordal graph with n vertices is at most n [7]. Thus, T has at most n_0 units per layer. Since each pair of hidden DBM layers forms a complete bipartite graph, it is straightforward to embed the two hidden layers of T into the first two layers of M (visualized in Fig. 1 d)).

Finally, setting all edge weights $\boldsymbol{\theta}_e$ of some edge e to the all-zero vector $\mathbf{0}$, implies that the corresponding edge potential $\psi_e(\boldsymbol{x}_e)$ is 1 for all choices of $\boldsymbol{x}_e$—the edge e is effectively removed from the undirected model. Thus, all edges which are not required to embed T into M can be removed. Together with Theorem 2, this shows that there exists a canonical weight vector $\boldsymbol{\theta}$ for the DBM which induces a probability mass that is arbitrarily close to the true pmf of $\boldsymbol{X}$. Hence, M is an universal approximator. □

Note, however, that this result is not contradictory to Theorem 1. Our theorem does not assume that observed and hidden units have the same state space. In our setting, the latent state space and thus, the complexity of the learned activation functions, is allowed to vary with the complexity of the input data. This is an important difference to ordinary feed-forwards architectures where the functional form of the activation functions is usually fixed.

While the theorem shows a constant dependence of the depth on the number of visible units, the dependence of the width is still linear. Inspecting the proof reveals that even the "vertical" worst-case embedding (Fig. 1 c)) of any DBT into the corresponding DBM can be realized as long as $L \geq 2n_0 - 1$—a linear worst-case depth. This suggests that no DBM must be deeper than $2n_0 - 1$ layers as long as the hidden units are expressive enough to cover the underlying clique potentials. Motivated by this observation, we state the following conjecture:

Conjecture 1 (The Depth of Deep Networks). DBMs with more than two hidden layers are only required if the underlying learning algorithm cannot find a shallow DBT embedding into the DBM structure.

Results on model compression suggest that shallow networks can be on par with state-of-the-art deep models [3, 1]. Such results rely on specialized training procedures, but finding a superior shallow solution directly might not be easy for the learning algorithm. Indeed, learning the weights of DBMs and other deep architectures suffers from various local minima—the 2-layer solution from Theorem 3 is only one of them. Which solution is learned eventually depends crucially on the weight initialization [8].

Another interesting fact is that the proof tells us how DBT learning is connected to classic and recent structure learning techniques.

Corollary 1 (Chow-Liu DBM). *Consider a data set $\mathcal{D} = \{(\boldsymbol{x}, \boldsymbol{h})^i : 1 \leq i \leq N\}$, sampled from the Deep Boltzmann Machine described in Theorem 3. Running the Chow-Liu structure estimation algorithm [5] on $\mathcal{D}$, and dropping all edges with uniform edge marginals and disconnected vertices reveals the DBT.*

By construction, the Chow-Liu tree is the pairwise undirected model that minimizes the Kullback-Leibler divergence to the actual joint pmf that generated the data. While we assume that the data was generated by a Boltzmann machine, we know that there is a Boltzmann tree which represents the exact same pmf. Thus, the Chow-Liu tree must be the DBT given N is large enough.

Corollary 2 (l_1-regularized DBM). *Consider a data set $\mathcal{D} = \{(\boldsymbol{x}, \boldsymbol{h})^i : 1 \leq i \leq N\}$, sampled from the Deep Boltzmann Machine described in Theorem 3.*

Algorithm 2: Learning the Deep Boltzmann Tree Weights

Require: Data set $\mathcal{D} = \{\boldsymbol{x}^1, \ldots, \boldsymbol{x}^N\}$, DBT $T = (V, E)$ from Alg. 1
1: Initialize $\boldsymbol{\theta} \leftarrow \mathbf{0}$
2: $\forall$ hidden unit u: initialize $\mathcal{X}_u \leftarrow \emptyset$
3: **for** training data $\boldsymbol{x}^i \in \mathcal{D}$ **do**
4: $\quad \boldsymbol{y} \leftarrow$ empty state ()
5: $\quad$ **for** hidden unit $u \in V$ **do**
6: $\quad\quad$ **for** vertex $v \in V(u)$ **do**
7: $\quad\quad\quad \boldsymbol{y} \leftarrow \boldsymbol{y} \circ \boldsymbol{x}_v^i$
8: $\quad\quad$ **end for**
9: $\quad\quad$ **if** $\boldsymbol{y} \notin \mathcal{X}_u$ **then**
10: $\quad\quad\quad \mathcal{X}_u \leftarrow \mathcal{X}_u \cup \{\boldsymbol{y}\}$
11: $\quad\quad$ **end if**
12: $\quad\quad \boldsymbol{h}(\boldsymbol{x}^i)_u \leftarrow \boldsymbol{y}$
13: $\quad$ **end for**
14: **end for**
15: **while** $\|\nabla(\boldsymbol{\theta})\| > 0$ **do**
16: $\quad \boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \frac{1}{2|E|}\nabla\ell(\boldsymbol{\theta})$
17: **end while**
18: **return** $\boldsymbol{\theta}$ // Optimal weights

Running the Elem-GM structure estimation algorithm [27] on $\mathcal{D}$ and dropping all fully disconnected vertices reveals the DBT.

The so called elementary estimator for graphical models (Elem-GM) is a regularization based structure learning technique. In contrast to the Chow-Liu tree, Elem-GM can output non-tree structures. The method performs l_1-regularization to identify unnecessary edges which are then excluded from the learned model. Since we know that many edges are actually unnecessary to recover the full joint pmf, we conclude that the DBT is an optimal solution to the Elem-GM problem given N is large enough.

3.1 Learning the DBT Weights

So far, we only discussed how to find the DBT. We will now explain how to estimate the DBT weights from data. Learning the parameters of a DBT factorizes into two phases: in the first phase, we have to find good initial values for the hidden units $\boldsymbol{h}$—this choice is crucial and failing to find good values implies inferior learning results. Moreover, phase one determines the state space $\mathcal{X}_u$ of each hidden units u. In phase two, numerical first-order optimization is applied to find a minimizer of ℓ. The problem in phase two is convex given any fixed hidden values from phase one. The learning procedure is provided in Alg. 2. Let us quickly go through it line-by-line. First, we initialize the weight vector and the hidden state spaces (lines 1 and 2). We then loop over all N training instances $\boldsymbol{x}^i$ (line 3). We initialize a new empty state (line 4) and recall from Theorem 2 that

each hidden unit u originates from a clique or separator vertex of the junction tree. Denote the set of visible units connected to the original clique or separator vertex by $V(u)$. In lines 6-8 we read the states of all visible units in $V(u)$ and construct a new hidden state $\boldsymbol{y}$. If that hidden state was never seen before (line 9), we create a new state in u's state space (line 10), and assign that new state to the hidden activation $\boldsymbol{h}(\boldsymbol{x}^i)_u$ that is associated with the current data point $\boldsymbol{x}^i$ (line 12). Lines 15 to 17 correspond to gradient descent.

Notice that we stop the optimization when the gradient's norm is zero. Since the objective function is convex, we will surely arrive at a global minimizer of ℓ given our learning rate is correct. Notice further that we set the learning rate to $1/(2|E|)$. This originates from the fact that gradient descent with learning rate $1/\mathcal{L}$ is guaranteed to converge to the next local minimum (which is also global due to convexity). Here $\mathcal{L}$ denotes the gradients Lipschitz constant. As we could not find the following result in the literature, we state it for completeness. A proof is provided in the supplementary material.

Lemma 1 (Lipschitz Continuous Gradient). *The gradient of any tree-structured, undirected model is Lipschitz continuous with constant* $\mathcal{L} = 2|E|$.

For simplicity, we state Alg. 2 as plain gradient descent method. In our experiments however, we use Nesterov-acceleration [17] to speed-up learning. More on gradient computation for exponential families can be found in [24].

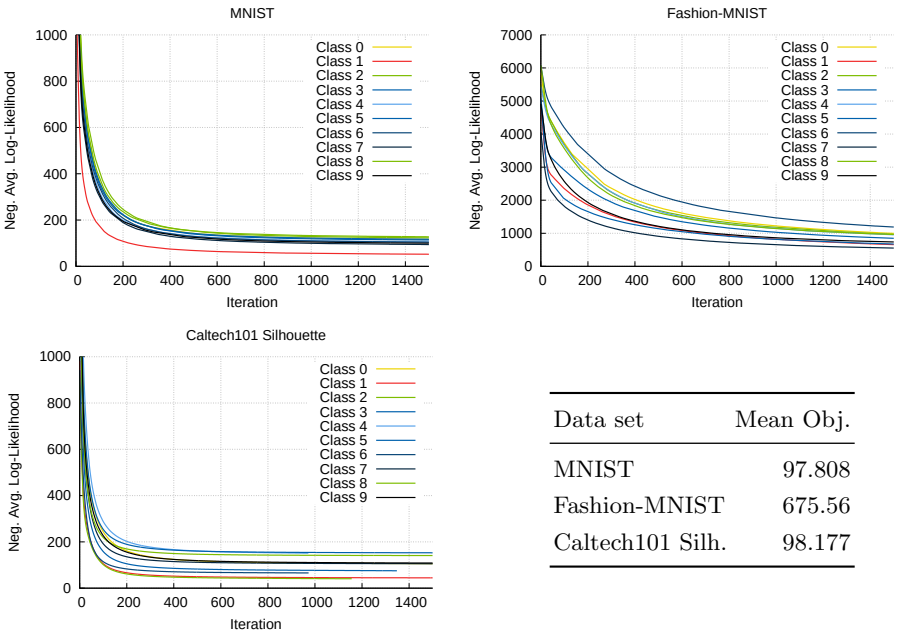
The proposed algorithm grows the hidden state space to cover joint realizations of the underlying chordal model. Note, however, that only clique states that actually appear in the data set are generated. This is in in contrast to the junction tree algorithm, whose runtime is always exponential in the size of the largest clique. However, if one cannot effort to grow the hidden state space as large as the data tells us, i.e., due to limited resources, we can assign some already known state. In that case, we suggest to iterate phase two and use the estimated model weights $\boldsymbol{\theta}$ to re-sample the hidden activation. Thus, performing an expectation-maximization procedure [6].

4 Experiments

We conduct a small set of experiments to provide a proof of concept of the generative capabilities of Deep Boltzmann Trees. The source code, and a docker image that contains everything which is required to repeat our experiments, are available for download (http://www.randomfields.org/dbt). To facilitate reproducability, we employ the following freely available benchmark data sets:

- MNIST (http://yann.lecun.com/exdb/mnist)
- Fashion-MNIST (https://github.com/zalandoresearch/fashion-mnist)
- Caltech101 Silhouette (https://people.cs.umass.edu/m̃arlin/data.shtml)

Numeric attributes (MNIST and Fashion-MNIST) are discretized via quantiles to contain at most 10 distinct states. The Caltech101 data contains various



Data set	Mean Obj.
MNIST	97.808
Fashion-MNIST	675.56
Caltech101 Silh.	98.177

Fig. 5. The training progress and mean neg. avg. log-likelihood of DBT learning on all benchmark data sets.

classes with very few data points (< 100). Thus, we took only the ten classes with most training instances. For each data set, we join the predefined training data and test data, and run Algorithms 1 and 2 until convergence to estimate the DBTs and their weights. Recall that Algorithm 1 requires a graphical structure as input. We run Chordalysis [26] to compute chordal conditional independence structures. Chordalysis allows to control the false discovery rate to get rid of spurious dependencies, which we set[2] to 0.05. The training progress and final mean objective function values are provided in Fig. 5. The plots show how the conditional likelihood of each class evolves during training. We see that the model achieves much lower neg. log-likelihoods on MNIST and Caltech101 than on Fashion-MNIST. Since the DBT itself is a universal estimator, we conclude that Fashion-MNIST does not contain enough data to allow a reliable estimation of the underlying conditional independence structure. Having a reasonable estimate of that structure is crucial for the DBT construction.

After learning, we perform Perturb-and-MAP sampling [19] to generate samples from the models. Due to large likelihood values, we expect that samples from the Fashion-MNIST model have rather low quality. Some resulting samples are shown in Figures 6 and 7. We see that the model produces crisp MNIST samples

[2] We have to stress that this is not a hyper-parameter of the DBT. Moreover, 0.05 is not "tuned" either as it is the default value in Chordalysis.

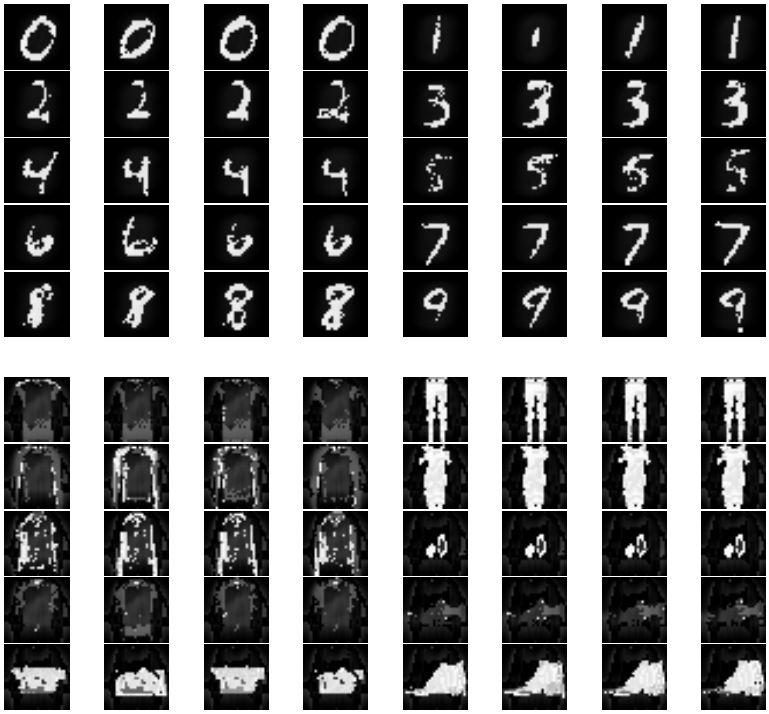


Fig. 6. Synthetic MNIST data (top) and synthetic Fashion-MNIST data (bottom), sampled from the DBT.

without mode collapse, i.e., they are not just noisy versions of the same number. Each class is able to generate multiple samples which look like different ways to write the particular number. As expected, the quality of Fashion-MNIST samples is rather low. The type of class, like pants, bag, shirt, shoe, or dress can be identified in most cases, but the resulting samples are close to mode collapse. The diversity of Caltech101 Silhouette samples is also low. However, this is already true for the original silhouette data. Of course, the model can only learn to generate diverse samples if the underlying data contains some diversity.

5 Conclusion

State-of-the-art results in various classification and synthetic data generation tasks are often achieved by deep learning. While the field of deep learning evolves fast, theoretical insights are rare. Moreover, many hyper-parameters have to be tuned in order to reach actual state-of-the-art performance. Driven by the wish for a better understanding of how depth improves a model, we studied the structure of DBMs. We discovered a new deep generative model, the Deep Boltzmann Tree, which can be learned from data without tuning a single hyper-parameter.

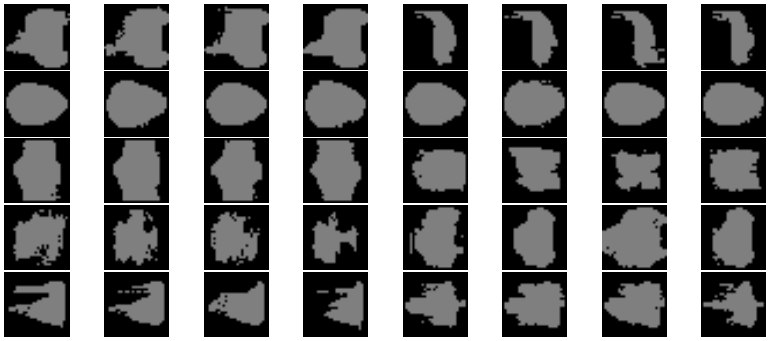


Fig. 7. Synthetic Caltech101 Silhouette data, sampled from the Deep Boltzmann Tree.

We proved that DBTs are universal approximators and showed connections to other structure learning methods. Experiments on benchmark data suggest, that high-quality synthetic data can be generated if the data set is large enough to allow for a reasonable estimation of the underlying conditional independence structure. Due to its tree structure, the DBT does not suffer from computational issues like the Deep Boltzmann Machine does. As a by-product, we discovered the best known bound on the depth of categorical DBMs and proposed a conjecture on why depth can improve a model in practice. Our results pave the way for several new research directions, including likelihood-based hybrid classification/generation models, and the consistent estimation of high-resolution image and audio data with theoretical guarantees.

Acknowledgments This research has been funded by the Federal Ministry of Education and Research of Germany as part of the competence center for machine learning ML2R (01S18038A).

References

1. Ba, J., Caruana, R.: Do deep nets really need to be deep? In: Advances in Neural Information Processing Systems (NIPS). pp. 2654–2662 (2014)
2. Bartlett, P.L., Foster, D.J., Telgarsky, M.J.: Spectrally-normalized margin bounds for neural networks. In: Advances in Neural Information Processing Systems (NIPS) 30. pp. 6241–6250 (2017)
3. Bucila, C., Caruana, R., Niculescu-Mizil, A.: Model compression. In: Int. Conf. on Knowledge Discovery and Data Mining (SIGKDD). pp. 535–541 (2006)
4. Bulatov, A., Grohe, M.: The complexity of partition functions. In: Automata, Languages and Programming, Lecture Notes in Computer Science, vol. 3142, pp. 294–306. Springer, Heidelberg, Germany (2004)
5. Chow, C., Liu, C.: Approximating discrete probability distributions with dependence trees. IEEE Transactions on Information Theory **14**(3), 462–467 (1968)
6. Dempster, A.P., Laird, M.N., Rubin, D.B.: Maximum likelihood from incomplete data via the em algorithm. Journal of the Royal Statistical Society. Series B (Methodological) **39**(1), 1–38 (1977)

7. Fulkerson, D.R., Gross, O.A.: Incidence matrices and interval graphs. Pacific Journal of Mathematics **15**(3), 835–855 (1965)
8. Glorot, X., Bengio, Y.: Understanding the difficulty of training deep feedforward neural networks. In: International Conference on Artificial Intelligence and Statistics (AISTATS). pp. 249–256 (2010)
9. Golowich, N., Rakhlin, A., Shamir, O.: Size-independent sample complexity of neural networks. In: Conference On Learning Theory (COLT). pp. 297–299 (2018)
10. Goodfellow, I.J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A.C., Bengio, Y.: Generative adversarial nets. In: Advances in Neural Information Processing Systems (NIPS). pp. 2672–2680 (2014)
11. Hammersley, J.M., Clifford, P.: Markov fields on finite graphs and lattices. Unpublished manuscript (1971)
12. He, K., Zhang, X., Ren, S., Sun, J.: Identity mappings in deep residual networks. In: uropean Conference on Computer Vision (ECCV). pp. 630–645 (2016)
13. Li, Y., Turner, R.E.: Gradient estimators for implicit models. In: International Conference on Learning Representations (ICLR) (2018)
14. Montavon, G., Braun, M.L., Müller, K.: Deep Boltzmann machines as feed-forward hierarchies. In: International Conference on Artificial Intelligence and Statistics (AISTATS). pp. 798–804 (2012)
15. Montúfar, G.: Deep narrow Boltzmann machines are universal approximators. In: International Conference on Learning Representations (ICLR) (2015)
16. Montúfar, G., Morton, J.: Discrete restricted Boltzmann machines. Journal of Machine Learning Research **16**, 653–672 (2015)
17. Nesterov, Y.: A method of solving a convex programming problem with convergence rate $O(1/k^2)$. Soviet Mathematics Doklady **27**(2), 372–376 (1983)
18. Neyshabur, B., Tomioka, R., Srebro, N.: Norm-based capacity control in neural networks. In: Conference on Learning Theory (COLT). pp. 1376–1401 (2015)
19. Papandreou, G., Yuille, A.L.: Perturb-and-MAP random fields: Using discrete optimization to learn and sample from energy models. In: IEEE International Conference on Computer Vision (ICCV). pp. 193–200 (2011)
20. Salakhutdinov, R., Hinton, G.E.: Deep Boltzmann machines. In: International Conference on Artificial Intelligence and Statistics (AISTATS). pp. 448–455 (2009)
21. Salakhutdinov, R., Larochelle, H.: Efficient learning of deep Boltzmann machines. In: Artificial Intelligence and Statistics (AISTATS). pp. 693–700 (2010)
22. Shazeer, N., Mirhoseini, A., Maziarz, K., Davis, A., Le, Q.V., Hinton, G.E., Dean, J.: Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. CoRR **abs/1701.06538** (2017)
23. Valiant, L.G.: The complexity of enumeration and reliability problems. SIAM Journal on Computing **8**(3), 410–421 (1979)
24. Wainwright, M.J., Jordan, M.I.: Graphical models, exponential families, and variational inference. Foundations and Trends in Machine Learning **1**(1–2), 1–305 (2008)
25. Warde-Farley, D., Bengio, Y.: Improving generative adversarial networks with denoising feature matching. In: Int. Conf. on Learning Representations (ICLR) (2017)
26. Webb, G.I., Petitjean, F.: A multiple test correction for streams and cascades of statistical hypothesis tests. In: International Conference on Knowledge Discovery and Data Mining (SIGKDD). pp. 1225–1264 (2016)
27. Yang, E., Lozano, A.C., Ravikumar, P.: Elementary estimators for graphical models. In: Advances in Neural Information Processing Systems (NIPS) 27. pp. 2159–2167 (2014)