On Recovering from Modeling Errors Using Testing Bayesian Networks

Haiying Huang[1] **Adnan Darwiche**[1]

Abstract

We consider the problem of supervised learning with Bayesian Networks when the used dependency structure is incomplete due to missing edges or missing variable states. These modeling errors induce independence constraints on the learned model that may not hold in the true, data-generating distribution. We provide a unified treatment of these modeling errors as instances of state-space abstractions. We then identify a class of Bayesian Networks and queries which allow one to fully recover from such modeling errors if one can choose Conditional Probability Tables (CPTs) dynamically based on evidence. We show theoretically that the recently proposed *Testing Bayesian Networks (TBNs),* which can be trained by compiling them into *Testing Arithmetic Circuits (TACs),* provide a promising construct for emulating this CPT selection mechanism. Finally, we present empirical results that illustrate the promise of TBNs as a tool for recovering from certain modeling errors in the context of supervised learning.

1. Introduction

Supervised learning has become very influential recently and stands behind most real-world applications of AI today. In supervised learning, one learns a function from labeled data, a practice that is now dominated by neural networks; see (Goodfellow et al., 2016; Hinton et al., 2006; Bengio et al., 2006; Ranzato et al., 2006). This particular use of neural networks is an example of *model-free* supervised learning to be contrasted with *model-based* supervised learning; see, e.g., (Bishop, 2013; Gao et al., 2018; Chen et al., 2020). The latter type feeds more information and assumptions into the learning process so should perform better in principle, let alone being more interpretable and also well-positioned to provide guarantees on the learned functions. Yet, model-based supervised learning has been outperformed by the model-free approach in many applications which has somewhat created a dilemma for AI (Darwiche, 2018).[1]

A major challenge for model-based supervised learning is the availability of accurate models. Consider Bayesian networks (Pearl, 1989), for example, which can be used for supervised learning as follows. Given a Bayesian network structure, one can learn its parameters to optimize a distribution for some query variable as a function of the values for some evidence variables. The used structure defines the space of functions that can be learned so if the structure is incorrect the space may fail to include the data-generating function. Hence, while the use of correct models can be very advantageous, the use of incorrect ones can be quite problematic. Moreover, it may be very difficult in some cases to build even approximate models as we may have very little understanding of the underlying phenomena. One can of course aim to learn the structure of a Bayesian network in addition to its parameters. This direction has received significant attention but poses substantial computational difficulties; see (Darwiche, 2009; Koller & Friedman, 2009; Murphy, 2012) for some comprehensive treatments on learning Bayesian networks and (Elidan et al., 2001; Kim & Leskovec, 2011; Elidan & Friedman, 2001) for some works that try to infer modeling errors from data.

We address the challenge of model-based supervised learning in this paper under the assumption that we have an incorrect Bayesian network structure. That is, the goal is to learn a function from labeled data using an incorrect structure while trying to recover from modeling errors that may have manifested. Our approach is based on two main insights, the first of which is motivated by earlier work on state-space abstractions (Wellman & Liu, 1994; Chang & Fung, 1990). In particular, we show that one can cast certain modeling errors (missing states and edges) as instances of such abstractions, which allows us to cast the recovery from modeling errors as a process of recovering from state-space abstractions. Our central contribution is a result which shows that, under certain conditions, one can fully recover from state-space abstractions if one is able to learn *dynamic parameters* for

[1]Computer Science Department, University of California, Los Angeles, USA. Correspondence to: Haiying Huang <hhaiying@ucla.edu>.

Proceedings of the 38th International Conference on Machine Learning, PMLR 139, 2021. Copyright 2021 by the author(s).

[1]Some use the term "model" more broadly, referring to a neural network as a (non-parametric) model.

the Bayesian network structure; that is, if one is able to learn a collection of parameters from which a set is selected during inference time based on evidence (function input). In other words, one has to learn multiple Bayesian networks which share the same (incomplete) structure and vary only across their parameters.[2]

While this may sound like a theoretical exercise, another main contribution in this paper shows how to realize it practically. This is where the second insight comes in, which is based on the recently introduced *Testing Bayesian Networks (TBNs)* (Choi & Darwiche, 2018; Choi et al., 2019). These models were introduced to address the expressiveness gap between functions induced by Bayesian networks (BNs) and those induced by neural networks. The first class of functions are multilinear (or their quotients) while the second class are universal function approximators (e.g., piecewise multilinear for neural networks with ReLU activation functions). A TBN is like a BN except that some of its nodes can select their CPTs dynamically based on evidence, which allows TBNs to induce universal function approximators like neural networks. What we show is that the evidence-based CPT selection process in TBNs can be used to implement the recovery approach that we develop theoretically. Beyond showing how our recovery proposal can be realized practically, this also provides a theoretical basis for the utilization of TBNs in recovering from modeling errors which has so far been only shown empirically (Shen et al., 2019).

This paper is organized as follows. We review Testing Bayesian Networks in Section 2 and discuss the casting of certain modeling errors as state-space abstractions in Section 3. We then present our approach for recovering from modeling errors in Section 4 and show how this approach can be implemented using Testing Bayesian Networks in Section 5. We present empirical results in Section 6 and close with some concluding remarks in Section 7. Proofs of theorems can be found in the supplementary material.

2. Testing Bayesian Networks

We use uppercase letters (e.g., X) to denote variables and lowercase letters (e.g., x) to denote their states. We use bold uppercase letters (e.g., $\mathbf{X}$) to denote sets of variables and bold lowercase letters (e.g., $\mathbf{x}$) to denote their instantiations.

A Bayesian Network (BN) over variables $\mathbf{X}$ is a pair (G, Θ) where G is a directed acyclic graph (DAG) over variables $\mathbf{X}$ and Θ is a set of Conditional Probability Tables (CPTs). Each node $X \in \mathbf{X}$ with parents $\mathbf{U}$ has one CPT $\Theta_{X|\mathbf{U}}$ which specifies a conditional distribution $\theta(X|\mathbf{u})$ for each instantiation $\mathbf{u}$ of parents $\mathbf{U}$. A BN represents a joint distribution over variables $\mathbf{X}$ (Pearl, 1989).

[2]Our approach can be viewed as reducing bias at the expense of increasing the variance.

Testing Bayesian Networks (TBNs) were introduced recently as a generalization of BNs by allowing CPTs to be chosen dynamically based on the evidence available at inference time (Choi & Darwiche, 2018; Choi et al., 2019). TBN nodes can be *regular* or *testing*. Consider a node X with parents $\mathbf{U}$. If X is regular, then it has a regular CPT $\Theta_{X|\mathbf{U}}$ as in a BN (root nodes of a TBN must be regular). If X is testing, it has a *testing CPT* $\Theta^t_{X|\mathbf{U}}$ which specifies multiple conditional distributions $\{\theta^k(X|\mathbf{u})\}_{k=1}^N$ for each parent instantiation $\mathbf{u}$. These distributions are associated with intervals $\{I^k_{\mathbf{u}}\}_{k=1}^N$ that partition the interval $[0,1]$.[3] At inference time, one distribution from $\{\theta^k(X|\mathbf{u})\}_{k=1}^N$ is selected based on the available evidence $\mathbf{e}$. In particular, distribution $\theta^k(X|\mathbf{u})$ is selected if the posterior $Pr_i(\mathbf{u}|\mathbf{e}_i)$ (explained next) belongs to the interval $I^k_{\mathbf{u}}$. This process is repeated for every parent instantiation $\mathbf{u}$ and the selected distributions are then assembled into a regular CPT for node X. When the CPT selection process concludes, the TBN is turned into a BN that is then used for inference under evidence $\mathbf{e}$. If the evidence changes, a new BN is selected and used for inference under the new evidence.

The BN selected under evidence $\mathbf{e}$ is constructed incrementally by selecting CPTs according to a topological order $\pi = X_1, X_2, \ldots, X_n$ of TBN nodes. Order π defines DAGs $G_1, \ldots, G_{n+1}$ where G_1 is empty and G_{i+1} is obtained by adding node X_i to G_i (so G_{n+1} is the original DAG G).

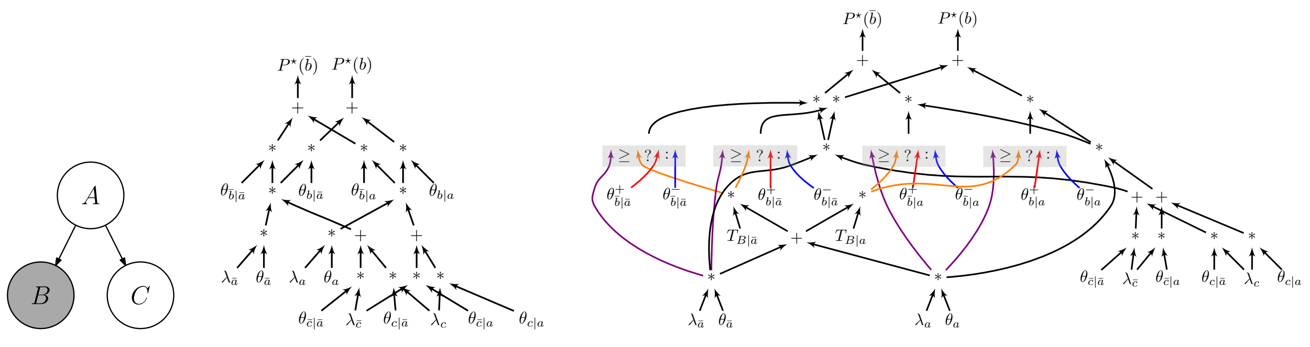
We start with the empty DAG G_1 and add root node X_1 with its regular CPT to yield BN (G_2, Θ_2) that induces distribution Pr_2. When adding node X_i for $i > 1$, we already have a BN (G_i, Θ_i) which induces distribution $Pr_i(.)$ and contains parents $\mathbf{U}_i$ of X_i. If node X_i is testing, we select its CPT based on the posterior $Pr_i(\mathbf{u}_i|\mathbf{e}_i)$ computed using BN (G_i, Θ_i). Otherwise, we use the regular CPT of node X_i.[4]

To completely define the semantics of TBNs, we now turn to the definition of $\mathbf{e}_i$ which we call the *selection evidence*. There are three definitions of this evidence, all of which ensure that evidence variables $\mathbf{E}_i$ are in DAG G_i (Choi & Darwiche, 2018; Choi et al., 2019) defined $\mathbf{e}_i$ as the subset of evidence $\mathbf{e}$ pertaining to ancestors of node X_i in DAG G. This was sufficient to show that TBNs are universal function approximators like neural networks. (Shen et al., 2019) defined $\mathbf{e}_i$ as the subset of evidence $\mathbf{e}$ that excludes evidence at or below testing nodes that are not ancestors of X_i. It further required order π to place non-testing nodes before testing nodes if possible,[5] which maximized the selection evidence

[3]The original definition of TBNs used $N = 2$ (Choi & Darwiche, 2018) but we generalize this definition to arbitrary N.

[4]If $Pr(.)$ is the distribution induced by the selected BN, then $Pr_i(\mathbf{u}_i|\mathbf{e}_i) = Pr(\mathbf{u}_i|\mathbf{e}_i)$ for all i; see the discussion on pruning Bayesian networks in Chapter 5 of (Darwiche, 2009).

[5]That is, if a regular node X_j is not a descendant of testing node X_i, then X_j must come before X_i in order π.



(a) BN/TBN (b) AC (c) TAC

Figure 1. A BN/TBN with a compiled AC and a compiled TAC for computing query $Pr(B|A,C)$ (after normalizing the AC/TAC outputs). Node B is testing so the TAC selects the parameters of node B based on the evidence on A and C (TAC input).

while keeping the selected BN invariant to the used topological order π. This increased the expressiveness of TBNs as it made CPT selection more dependent on evidence $\mathbf{e}$. In this work, we also require order π to satisfy this property but define $\mathbf{e}_i$ as the subset of evidence $\mathbf{e}$ pertaining to DAG G_i. Our definition increases the selection evidence and TBN expressiveness even further, but it makes the selected BN sensitive to the used order π. This additional expressiveness of TBNs is needed for recovering form a certain class of modeling errors as we show in Section 5.

As mentioned earlier, we are interested in a supervised learning setting where the goal is to learn a function that computes query $Pr(Q|\mathbf{e})$ based on a BN (G,Θ) (this function can be used, for example, to classify Q based on evidence $\mathbf{e}$). This can be realized by compiling the BN into an Arithmetic Circuit (AC) (Darwiche, 2003), whose parameters Θ are then learned using gradient descent as shown in (Darwiche, 2020); see Figure 1b. However, since DAG G may be incomplete due to missing edges or states, we will instead compile a TBN (G,Θ^t) into a Testing Arithmetic Circuit (TAC) (Choi & Darwiche, 2018; Choi et al., 2019) and learn its parameters Θ^t using gradient descent; see Figure 1c. This effectively learns a set of BNs, one of which is selected based on evidence $\mathbf{e}$ when computing the query $Pr(Q|\mathbf{e})$. As we show theoretically in Section 5 and empirically in Section 6, this can help in recovering from some of the modeling errors introduced when building DAG G.

3. Modeling Errors as State-Space Abstractions

We now show that missing certain edges or states when building a Bayesian network structure can be cast and analyzed in terms of state-space abstraction. This will aid us later when discussing recovery from such modeling errors.

To *abstract* the state-space of node X in a BN is to *merge* some states of X. This technique was explored in (Chang & Fung, 1990; Wellman & Liu, 1994) and can be used, for example, to speed up inference. Following these works, we call such a node X an *abstracted node.* We also refer to the original states of X as *elementary* and the new states as *superstates.* A state-space abstraction can be specified by a pair $\mathsf{a} = (sup, elm)$ where $sup(x)$ is the superstate of elementary state x and $elm(x')$ are the elementary states of superstate x'. We define these functions for all nodes so if node X is not abstracted, we have $sup(x) = x$ and $elm(x) = \{x\}$. We also extend state-space abstractions to sets of variables as follows: $sup(x_1, \ldots, x_n) = sup(x_1), \ldots, sup(x_n)$ and $elm(x'_1, \ldots, x'_n) = elm(x'_1) \times \ldots \times elm(x'_n)$. We often write $x \in x'$ as a shorthand for $x \in elm(x')$.

We will attach a state-space abstraction $\mathsf{a} = (sup, elm)$ as a superscript to various objects as needed. For example, $\mathbf{x}^{\mathsf{a}}$ is an abstracted state (instantiation) of variables $\mathbf{X}$, G^{a} is a DAG G with some abstracted nodes, and $Pr^{\mathsf{a}}(.)$ is a distribution over the abstracted space of distribution $Pr(.)$.

One ideally wants to preserve consistency across distributions when abstracting nodes of a BN.

Definition 1. *Distributions $Pr(.)$ and $Pr^{\mathsf{a}}(.)$ over variables $\mathbf{X}$ are said to be 'consistent' iff $Pr^{\mathsf{a}}(\mathbf{x}^{\mathsf{a}}) = \sum_{\mathbf{x}\in\mathbf{x}^{\mathsf{a}}} Pr(\mathbf{x})$.*

Consistency ensures that we can use the abstracted distribution Pr^{a} to answer any query about Pr as long as the query does not reference abstracted nodes. The key question is whether this is possible when abstracting some nodes of a BN (G,Θ). That is, can we revise CPTs Θ into some new CPTs Θ^{a} of the abstracted DAG G^{a} so that the two BNs are consistent? This would be significant if possible as it implies that we can recover from state-space abstractions (and some modeling errors) by potentially learning the proper CPTs Θ^{a}. This is generally not possible though. To see this, consider the BN in Figure 2 and suppose node X has three states x_1, x_2 and x_3. Any distribution induced by this BN must find Y independent of U given X and V. If we

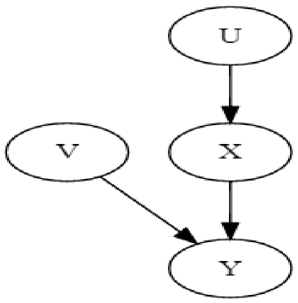


Figure 2. A Bayesian network structure.

abstract node X so it has superstates x_1 and $\{x_2, x_3\}$, then any distribution induced by the abstracted BN must also satisfy this independence, including when $X = \{x_2, x_3\}$. Yet this independence may not hold in the BN distribution: knowing the state of V and knowing only that the state of X is in $\{x_2, x_3\}$ may not make Y independent of U.

The following local but strong condition does, however, guarantee the sought consistency.

Definition 2. *A state-abstraction* $\mathsf{a} = (sup, elm)$ *is 'harmless' for a BN* (G, Θ) *if for every node* X *with parents* $\mathbf{U}$ *we have* $\theta(X|\mathbf{u}) = \theta(X|\mathbf{u}^\star)$ *when* $sup(\mathbf{u}) = sup(\mathbf{u}^\star)$.

This condition says that once we know a superstate of parents $\mathbf{U}$, then X becomes independent of the elementary states of $\mathbf{U}$. This somewhat generalizes the classical notion of context-specific independence (Boutilier et al., 1996).

Theorem 1. *If* a *is a harmless state-abstraction for BN* (G, Θ)*, then there is a BN* $(G^\mathsf{a}, \Theta^\mathsf{a})$ *which induces a distribution* $Pr^\mathsf{a}(.)$ *that is consistent with* $Pr(.)$.

The condition of Definition 2 is quite strong but the notion of consistency is also quite strong as it allows one to correctly compute any query using the abstracted BN as long as the query does not reference abstracted nodes. As we show in Section 4, one can significantly weaken this condition if one is only interested in recovering specific queries, which is all that that one needs in a supervised learning context.

A modeling error that misses some states of variables can be naturally cast as a state-abstraction of these variables. We now show that missing certain edges can also be cast in these terms. Our formulation uses the following definition.

Definition 3. *Let* G *be a DAG which results from adding some edges to DAG* G' *and abstracting some of its nodes. Then* G *is 'reducible' to* G' *iff for every BN* (G, Θ) *there is a BN* (G', Θ') *where distributions* Pr *and* Pr' *are consistent.*

Recall that our interest is to learn distributions through learning the parameters of a Bayesian network structure. This definition says that any distribution (of interest) which can learned using DAG G can also be learned using DAG G'.

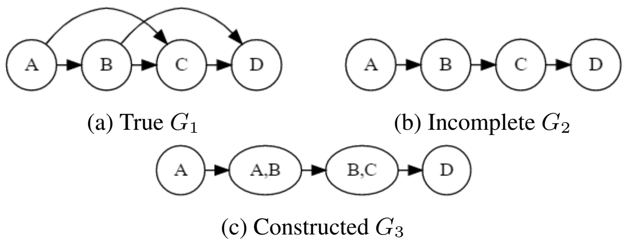


Figure 3. DAG G_1 results from adding edges and abstracting nodes in DAG G_3. DAGs G_2 and G_3 have the same topology.

Suppose now that we have a true structure G_1 (data-generating model) and a structure G_2 (used for learning from data) that misses some edges from G_1. We will next show how to construct a third structure G_3 that has the same topology as G_2 but more states than G_2 while guaranteeing that G_1 is reducible to G_3; that is, using G_3 does not preclude one from learning the data-generating model G_1. This will then show that if one can recover from having used G_2 instead of G_3 (state-space abstraction) then one can recover from having used G_2 instead of G_1 (missing edges).

For a concrete example, consider Figure 3 which depicts a true structure G_1 and a structure G_2 that misses two edges in G_1. We also have structure G_3 which has the same topology as G_2 except that we added more states to variables B and C: the state-space of B was multiplied by the state-space of A and the state-space of C was multiplied by the state-space of B. The true structure G_1 is reducible to structure G_3. Moreover, structure G_2 is the result of abstracting some nodes of G_3. Hence, if one can recover from the state-space abstraction that led to G_2 (from G_3) then one can recover from the missing edges that led to G_2 (from G_1).

For missing edges, we only treat what we call *high-order edges* $U \rightarrow X$ for which there exists some directed path from U to X other than edge $U \rightarrow X$.[6] We next show how the auxiliary DAG G_3 is constructed as in Figure 3.

Definition 4. *To 'harmlessly delete' high-order edge* $U \rightarrow X$ *from DAG* G*: remove edge* $U \rightarrow X$ *from* G*; choose a directed path* $U \rightarrow Z_1 \rightarrow \cdots \rightarrow Z_n \rightarrow X$*; and finally expand the state-space of each node* Z_i *to the Cartesian product of its original state-space and the state-space of* U.

Theorem 2. *If DAG* G_2 *is the result of harmlessly deleting high-order edges from DAG* G_1*, then* G_1 *is reducible to* G_2.

The reduction of high-order HMMs to first-order ones (He, 1988) is a special case of this theorem as one can obtain the latter by harmlessly deleting high-order edges from the former. Hence, one implication of our upcoming treatment is that TBNs can be used to recover from a modeling error

[6]Our treatment can be extended to other types of edges but this requires a discussion that is beyond the scope of this paper.

that uses a first-order HMM instead of a high-order one. (Shen et al., 2019) provided some preliminary empirical results to this effect but our upcoming results are analytic.

4. Recovering from State-Space Abstractions

We now address the question of revising the CPTs of a BN after abstracting some of its nodes, where the goal is to maintain the ability to compute some queries using the abstracted BN. We start with the proposal of (Wellman & Liu, 1994) which we call the *soft policy*. For a node X with parents $\mathbf{U}$ this policy defines the revised CPTs as follows:

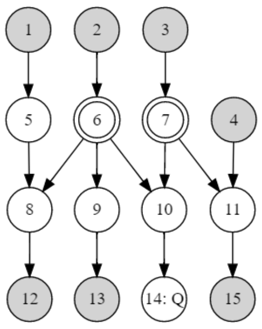
$$\begin{aligned}\theta^{\mathsf{a}}(x^{\mathsf{a}}|\mathbf{u}^{\mathsf{a}}) &= Pr(x^{\mathsf{a}}|\mathbf{u}^{\mathsf{a}}) \\ &\triangleq Pr(x \in x^{\mathsf{a}}|\mathbf{u} \in \mathbf{u}^{\mathsf{a}}) \\ &= \frac{\sum_{x \in x^{\mathsf{a}}, \mathbf{u} \in \mathbf{u}^{\mathsf{a}}} \theta(x|\mathbf{u}) Pr(\mathbf{u})}{\sum_{\mathbf{u} \in \mathbf{u}^{\mathsf{a}}} Pr(\mathbf{u})} \qquad (1)\end{aligned}$$

If node X and its parents are not abstracted, we get $\theta^{\mathsf{a}}(x^{\mathsf{a}}|\mathbf{u}^{\mathsf{a}}) = \theta(x|\mathbf{u})$ so the CPT of X is not revised. Consider Figure 2 and suppose that node X is the only abstracted node. Then only the CPTs of X and its child Y are revised. (Wellman & Liu, 1994) proposed another, less expensive CPT revision policy, which we call the *average policy:*

$$\theta^{\mathsf{a}}(x^{\mathsf{a}}|\mathbf{u}^{\mathsf{a}}) \triangleq \sum_{x \in x^{\mathsf{a}}} \underset{\mathbf{u} \in \mathbf{u}^{\mathsf{a}}}{\text{Average}}\, \theta(x|\mathbf{u}) \qquad (2)$$

While these policies may appear natural, the abstracted distributions Pr^{a} they produce are generally not consistent with Pr. This should not be surprising for the reasons we discussed earlier: aggregating states induces additional independencies that may not hold in the original BN. We will address this challenge based on two main insights. First, we will aim to only preserve $Pr(Q|\mathbf{e})$ for some query node Q and evidence variables $\mathbf{E}$; as mandated by supervised learning. Second, we will revise CPTs dynamically based on the available evidence $\mathbf{e}$ (recall how TBNs select CPTs). We will next identify a particular class of modeling errors and queries for which these insights will lead to full recovery.

Our treatment assumes that we are recovering from having abstracted a *polytree* BN; that is, a BN in which two nodes can be connected through at most one (undirected) path (Pearl, 1989). While a polytree has a restricted structure, recall that it may be the result of having missed edges so the data-generating BN does not have to be a polytree. That is, we are treating modeling errors that lead to a polytree structure not ones in which the true BN is a polytree. This includes, for example, learning using a first-order HMM (polytree) when the data-generating model is a high-order HMM (not a polytree). The queries $Pr(Q|\mathbf{e})$ we shall preserve are ones for which evidence variables $\mathbf{E}$ are not abstracted and the query node Q is a descendant of abstracted nodes (hence, node Q is not abstracted either).



(a) Polytree BN

Node	$\mathbf{e}^{\pi}_{X}$
8	1,2
9	1,2,12
10	1,2,3,4,12,13,15
11	3,4

(b) Evidence for children of abstracted nodes.

Figure 4. Abstracted nodes are double-circled. Evidence nodes are shaded. The query node is 14. The Q-ordering is $\pi = 1, 2, 3, 5, 6, 7, 4, 8, 12, 9, 13, 11, 15, 10, 14$.

Our approach amounts to the following. For each node X with parents $\mathbf{U}$, we identify some evidence $\mathbf{e}^{\pi}_{X}$ (explained later) and set the CPT of X to $Pr(x^{\mathsf{a}}|\mathbf{u}^{\mathsf{a}}, \mathbf{e}^{\pi}_{X})$. The abstracted polytree with these revised CPTs will then preserve the query $Pr(Q|\mathbf{e})$. Hence, the CPTs of an abstracted polytree are evidence-specific: if we change evidence $\mathbf{e}$ we have to reassign these CPTs accordingly. In Section 5, we show how this can be realized using TBNs.

To fully specify our approach, all we need now is to define evidence $\mathbf{e}^{\pi}_{X}$ which requires an ordering π of the polytree nodes that depends on query node Q. There may be more than one ordering π that satisfies the properties we shall state, each leading to a different evidence $\mathbf{e}^{\pi}_{Y}$. Yet, our main result in this section will show that any such ordering is guaranteed to preserve the query $Pr(Q|\mathbf{e})$.

Definition 5. *Let G be a polytree, Q be a node in G and U be some ancestor of Q. A child X of U will be called a 'Q-child' of U iff $X = Q$ or X is an ancestor of Q.*

If a node has a Q-child then it must be unique. This follows from G being a polytree. Figure 4a depicts a polytree where node 10 is the unique Q-child of abstracted nodes 6 and 7.

Definition 6. *Let G be a polytree and Q be a node in G. A 'Q-ordering' π of G is a total ordering of the nodes in G that places each Q-child after its siblings.*

A Q-ordering always exists. This follows because if a node X is the Q-child of two of its parents, then X must be the only common child of these parents so the children of these parents can be ordered independently. Consider the polytree in Figure 4a where node 14 is the query node. The following are two Q-orderings for this polytree.

$$\begin{aligned}\pi_1 &= 1, 2, 3, 5, 6, 7, 4, 8, 12, 9, 13, 11, 15, 10, 14 \\ \pi_2 &= 1, 2, 3, 5, 6, 7, 4, 9, 13, 11, 15, 8, 12, 10, 14\end{aligned}$$

We are now ready to define the evidence $\mathbf{e}^{\pi}_{X}$ which utilizes the following standard notation for partitioning evidence $\mathbf{e}$ across a polytree edge $U \rightarrow X$ (Pearl, 1989).

$\mathbf{e}^{+}_{UX}$: evidence on the U-side of edge $U \rightarrow X$
$\mathbf{e}^{-}_{UX}$: evidence on the X-side of edge $U \rightarrow X$
$\mathbf{e}^{-}_{X}$: evidence on the descendants of node X
$\mathbf{e}^{+}_{X}$: evidence on the non-descendants of node X

Definition 7. *Let G be a polytree with query node Q, evidence $\mathbf{e}$ and Q-ordering π. If node X has a parent U that is an ancestor of Q, we define*

$$\mathbf{e}^{\pi}_{X} \triangleq \mathbf{e}^{+}_{X} \setminus \{\mathbf{e}^{-}_{UY} \mid Y \textit{ is a child of } U \textit{ and } \pi(Y) > \pi(X)\}.$$

If node X has multiple parents U that are ancestors of Q, then X must be the Q-child of both parents and hence ordered last among its siblings. This leads to $\mathbf{e}^{\pi}_{X} = \mathbf{e}^{+}_{X}$ regardless of which parent U we pick in the above definition.

The definition of evidence $\mathbf{e}^{\pi}_{X}$ depends on the Q-ordering π though. Consider the polytree in Figure 4a and the two Q-orderings π_1 and π_2 shown earlier. For node $X = 8$ in Figure 4a, we have $\mathbf{e}^{\pi_1}_{X} = \{1, 2\}$ and $\mathbf{e}^{\pi_2}_{X} = \{1, 2, 13\}$. However, if two Q-orderings π_1 and π_2 agree on how they order the siblings of Q-children, then $\mathbf{e}^{\pi_1}_{X} = \mathbf{e}^{\pi_2}_{X}$.

We are now ready to define our *evidence-based policy* for assigning CPTs after abstracting nodes.

Definition 8. *Consider a polytree BN with distribution Pr. For query $Pr(Q|\mathbf{e})$, Q-ordering π and state-space abstraction a, assign CPTs for each node X with parents $\mathbf{U}$ as follows. If $\mathbf{U}$ contains abstracted nodes, then $\theta^{\mathsf{a}}(x^{\mathsf{a}}|\mathbf{u}^{\mathsf{a}}) = Pr(x^{\mathsf{a}}|\mathbf{u}^{\mathsf{a}}, \mathbf{e}^{\pi}_{X})$ else $\theta^{\mathsf{a}}(x^{\mathsf{a}}|\mathbf{u}^{\mathsf{a}}) = Pr(x^{\mathsf{a}}|\mathbf{u}^{\mathsf{a}})$.*

The following is our main result showing that the above CPT assignment policy will preserve the query of interest.

Theorem 3. *Consider a polytree BN with distribution Pr, a query $\Pr(Q|\mathbf{e})$ and a state-space abstraction a where abstracted nodes are ancestors of Q and do not overlap with $\mathbf{E}$. If Pr^{a} is the distribution of the BN abstracted according to Definition 8, then $Pr^{\mathsf{a}}(Q|\mathbf{e}) = Pr(Q|\mathbf{e})$.*

For a concrete example, consider the polytree BN $E \leftarrow R \rightarrow Q$ with the following CPTs (variables E and Q are binary, variable R has three states):

R	Θ_R
r_1	0.2
r_2	0.3
r_3	0.5

R	E	$\Theta_{E\|R}$
r_1	e	0.2
r_1	$\bar{e}$	0.8
r_2	e	0.5
r_2	$\bar{e}$	0.5
r_3	e	0.7
r_3	$\bar{e}$	0.3

R	Q	$\Theta_{Q\|R}$
r_1	q	0.6
r_1	$\bar{q}$	0.4
r_2	q	0.1
r_2	$\bar{q}$	0.9
r_3	q	0.9
r_3	$\bar{q}$	0.1

This BN induces the following query, which maps values of evidence variable E to distributions on query variable Q:

E	Q	$Pr(Q\|E)$
e	q	0.656
e	$\bar{q}$	0.344
$\bar{e}$	q	0.535
$\bar{e}$	$\bar{q}$	0.465

Suppose that we abstract node R by merging its states r_2 and r_3 so it now has two superstates r_1 and r_{23}. Using Q-ordering $\pi = R, E, Q$ we get $\mathbf{E}^{\pi}_{E} = \{\}$ and $\mathbf{E}^{\pi}_{Q} = \{E\}$ so the selection evidence for node E is empty. According to Definition 8, the abstracted BN will have one CPT for each of variables R and E:

R^{a}	$\Theta_{R^{\mathsf{a}}}$
r_1	0.2
r_{23}	0.8

R^{a}	E	$\Theta_{E\|R^{\mathsf{a}}}$
r_1	e	0.2
r_1	$\bar{e}$	0.8
r_{23}	e	0.625
r_{23}	$\bar{e}$	0.375

However, variable Q will have two CPTs, one of which is chosen depending on the available evidence on variable E:

R^{a}	Q	$\Theta^1_{Q\|R^{\mathsf{a}}}$
r_1	q	0.6
r_1	$\bar{q}$	0.4
r_{23}	q	0.5
r_{23}	$\bar{q}$	0.5

R^{a}	Q	$\Theta^2_{Q\|R^{\mathsf{a}}}$
r_1	q	0.6
r_1	$\bar{q}$	0.4
r_{23}	q	0.66
r_{23}	$\bar{q}$	0.34

CPT $\Theta^1_{Q|R^{\mathsf{a}}} = Pr(Q|R^{\mathsf{a}}, e)$ is used under evidence e and CPT $\Theta^2_{Q|R^{\mathsf{a}}} = Pr(Q|R^{\mathsf{a}}, \bar{e})$ is used under evidence $\bar{e}$. One can verify that this abstracted BN will induce the same query as the original BN, $Pr^{\mathsf{a}}(Q|E) = Pr(Q|E)$.

5. Evidence-Based Recovery Using TBNs

As discussed in Section 3, state-space abstraction encapsulates modeling errors in the form of missing states and edges. Theorem 3 tells us that we can recover queries $Pr(Q|\mathbf{e})$ from state-space abstractions and hence from these modeling errors by choosing the CPTs of a Bayesian network based on evidence $\mathbf{e}$ (under the stated conditions). Focusing on a particular query is not that restrictive as it is the mark of supervised learning where the goal is to learn functions, such as the mapping from evidence $\mathbf{e}$ into a posterior $Pr(Q|\mathbf{e})$. On the other hand, the selection of CPTs based on evidence may seem like a theoretical exercise except that it is not. The recently introduced TBN does exactly this (Choi & Darwiche, 2018; Choi et al., 2019). As discussed in Section 2, a TBN is a collection of BNs that share the same structure but differ only in their CPTs. One of these BNs is selected based on evidence and then used to answer queries where the selection process amounts to choosing the BN CPTs. This raises the following question: Can a TBN emulate the CPT selection policy discussed in Definition 8 which guarantees full recovery from state-space abstraction under some conditions? The multiple CPTs associated with a node in a TBN are meant to be learned from labeled data using methods such as gradient descent. This raises another question: Can the learned CPTs of a TBN reach the promise of Theorem 3? We tackle the first question next while leaving the second question to the next section.

Recall from Section 2 that a TBN has two types of nodes: regular and testing. A testing node X with parents $\mathbf{U}$ has a set distributions $\theta^1(X|\mathbf{u}), \ldots, \theta^N(X|\mathbf{u})$ for each parent instantiation $\mathbf{u}$, in addition to a partition $I^1_{\mathbf{u}}, \ldots, I^N_{\mathbf{u}}$ of the interval $[0, 1]$. Before answering a query under evidence $\mathbf{e}$, the TBN computes a posterior p on parent instantiation $\mathbf{u}$ and then chooses distribution $\theta^i(X|\mathbf{u})$ if $p \in I^i_{\mathbf{u}}$. We next discuss how this selection mechanism can be used to emulate the CPT selection process of Definition 8. *The main insight is to make each child of an abstracted node a testing node, which allows its CPT to be chosen based on evidence.*

One subtlety with TBNs is that parent posteriors cannot be computed before certain CPTs have been selected. In particular, before computing the parents posterior for node X one needs to at least select the CPTs for testing nodes that are ancestors of X. As discussed in Section 2, this is accomplished by using a topological ordering $X_1, \ldots, X_n$ of TBN nodes to create a sequence of DAGs $G_1, \ldots, G_{n+1}$, where DAG G_{i+1} results from adding node X_i to DAG G_i. The CPT of a testing node X_i is selected when adding node X_i to DAG G_i since at that point the CPTs of all nodes in G_i have been selected. Moreover, parents $\mathbf{U}_i$ of X_i are already in G_i so we can compute $Pr_i(\mathbf{U}_i \mid \mathbf{e}_i)$ using the distribution $Pr_i(.)$ of BN G_i. By the time variable X_n is added, we get BN G_{n+1} which is used to answer queries under evidence $\mathbf{e}$.

Two observations are relevant to the quest of this section. First, the evidence $\mathbf{e}_i$ used to compute the posterior $Pr_i(\mathbf{U}_i \mid \mathbf{e}_i)$ is the subset of evidence $\mathbf{e}$ pertaining to BN G_i so selecting the CPT of node X_i cannot use all available evidence $\mathbf{e}$. Second, the selection of this CPT is based only on the value of posterior $Pr_i(\mathbf{U}_i \mid \mathbf{e}_i)$. Do these restrictions limit the ability to emulate the CPT selection process of Definition 8? As we show next, the first restriction does not but the second restriction may, yet we identify a condition under which it does not.

If the topological ordering $\pi = X_1, \ldots, X_n$ used to select the CPTs of a TBN satisfies the condition of Definition 6, then evidence $\mathbf{e}_i$ will be a superset of evidence $\mathbf{e}^\pi_{X_i}$ needed by Definition 8. Moreover, one can always construct such a topological ordering π.[7] This settles the first question.

As to the second question, to fully emulate Definition 8, one may need as many intervals $I^1_{\mathbf{u}_i}, \ldots, I^N_{\mathbf{u}_i}$ as the number of instantiations for evidence variables $\mathbf{E}^\pi_{X_i}$ which is not possible practically. Moreover, two distinct instantiations of variables $\mathbf{E}^\pi_{X_i}$ may lead to the same posterior $Pr_i(\mathbf{U}_i \mid \mathbf{e}_i)$ so a TBN cannot fully emulate Definition 8. Even if it can, this would not be realized if we bound the number of intervals N and, hence, the number of CPTs associated with a testing node. However, we present experiments in the next section which show that a TBN can recover queries accurately even with a relatively small N.

[7]We visit TBN nodes parents before children. When there is a tie, we visit non-testing nodes before testing nodes. When there is a tie among testing nodes, we visit them according to Definition 6.

	BN		TBN			
Size	AVG	SOFT	N=2	N=10	N=20	N=50
10	0.143	0.0352	0.0271	0.0196	0.0106	**0.0027**
20	0.181	0.0327	0.0289	0.0215	0.0094	**0.0075**
30	0.285	0.0419	0.0346	0.0249	0.0211	**0.0154**
40	0.251	0.0306	0.0262	0.0228	0.0210	**0.0154**
50	0.264	0.0521	0.0446	0.0254	0.0229	**0.0195**

Table 1. Conditional KL-divergance between handcrafted BN/TBN and true BN. Each data point is an average over 50 trials.

For the first set of experiments, we *calculate* the TBN CPTs using the same procedure for selecting TBN CPTs except that we now *construct* the space of CPTs instead of *selecting* CPTs from a given space. We first fix a number of intervals N. When adding node X_i to BN G_i, we create buckets $\mathcal{E}^1_{\mathbf{u}_i} = \emptyset, \ldots, \mathcal{E}^N_{\mathbf{u}_i} = \emptyset$ for every instantiation $\mathbf{u}_i$ of X_i's parents. We then enumerate every possible instantiation $\mathbf{e}^\pi_{X_i}$ of evidence variables $\mathbf{E}^\pi_{X_i}$. We compute the posterior $p = Pr_i(\mathbf{u}_i \mid \mathbf{e}_i)$ and add instantiation $\mathbf{e}^\pi_{X_i}$ to bucket $\mathcal{E}^k_{\mathbf{u}_i}$ if $p \in I^k_{\mathbf{u}_i}$. After processing all evidence instantiations $\mathbf{e}^\pi_{X_i}$, we associate the following distribution with interval $I^k_{\mathbf{u}_i}$:

$$\begin{aligned} & Pr(X_i \mid \mathbf{u}_i, \mathbf{e}^\pi_{X_i} \in \mathcal{E}^k_{\mathbf{u}_i}) \\ &= \eta \sum_{\mathbf{e}^\pi_{X_i} \in \mathcal{E}^k_{\mathbf{u}_i}} Pr(X_i \mid \mathbf{u}_i, \mathbf{e}^\pi_{X_i}) Pr(\mathbf{u}_i \mid \mathbf{e}^\pi_{X_i}) Pr(\mathbf{e}^\pi_{X_i}), \end{aligned} \tag{3}$$

where η is a normalizing constant. This method, which we call the *TBN policy*, divides the evidence space into N buckets $\mathcal{E}^1_{\mathbf{u}_i}, \ldots, \mathcal{E}^N_{\mathbf{u}_i}$ based on which interval the parents posterior falls into. It then computes a weighted average of the distributions in each bucket. If some bucket $\mathcal{E}^k_{\mathbf{u}_i}$ has more than one distribution, we say we have a *CPT collision*. In the absence of CPT collisions, the TBN policy fully emulates the CPT selection process of Definition 8. We can view the number of intervals N as a tradeoff between recovery quality and memory cost. When $N = 1$, the TBN policy degenerates to the soft policy (Equation 1). As N approaches the size of the evidence space, the TBN will fully recover the query $Pr(Q|\mathbf{e})$ if no CPT collision happens.

6. Experiments

We now empirically evaluate the techniques we proposed in Sections 4 and 5. We consider two tasks: *model compression* and *supervised learning*. Both tasks try to recover a function $Pr(Q|\mathbf{E})$ generated by a true BN (G, Θ) but using an incomplete BN structure G' that may be the result of a modeling error. The recovery is accomplished using a TBN with structure G'. In the first task, we handcraft TBN parameters (CPTs and intervals) based on the true BN and the policy of Section 5 (see Equation 3). The goal is to assess how well can a TBN reach the promise of full recovery from modeling errors using a finite number of intervals. In

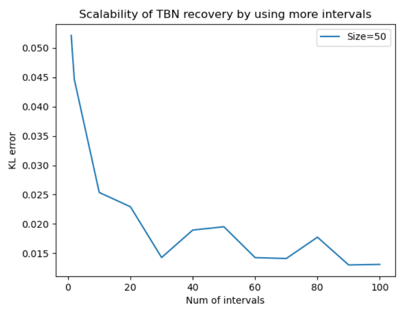


Figure 5. Recovery using handcrafted TBNs as a function of the number of equally-spaced intervals N. Each data point is an average over 50 trials for polytrees with 50 nodes.

the second task, we assess whether learning TBN parameters, instead of handcrafting them, can potentially reach the promise of full recovery by comparing the learned function with the handcrafted and true ones. This experiment is somewhat limited in scope since learning TBN parameters is still in its infancy and we do not yet have learning algorithms that are robust enough. This task is quite challenging as we are trying to recover from modeling errors based on data and without knowing the nature of these modeling errors.

We use the conditional KL-divergence to measure the similarity between the true function $Pr(Q|\mathbf{E})$ and the learned or handcrafted one $Pr'(Q|\mathbf{E})$:

$$KL(Pr||Pr') = \sum_{\mathbf{e}} Pr(\mathbf{e}) \sum_{q} Pr(q|\mathbf{e}) \log \frac{Pr(q|\mathbf{e})}{Pr'(q|\mathbf{e})}.$$

6.1. Model Compression

For this experiment, we randomly generate polytree BNs and queries $Pr(Q|\mathbf{E})$. We simulate a modeling error by abstracting some nodes of the polytree to yield an incomplete structure G'. We then parameterize G' according to the soft policy (Equation 1) and average policy (Equation 2) to yield two BNs that we call BN_SOFT and BN_AVG. We also parameterize structure G' according to the TBN policy (Equation 3) to yield a TBN. We finally compare the three functions $Pr'(Q|\mathbf{E})$ induced by the two BNs and TBN. We do this by measuring their proximity to the ground-truth function $Pr(Q|\mathbf{E})$ induced by the true BN model.

We use the method of (Ide & Cozman, 2002) to generate random polytree structures. We randomly choose a leaf node to be query Q and 20% of the nodes to be evidence nodes $\mathbf{E}$. We assign 5 states to all ancestors of Q and 2 states to other nodes. CPTs are generated by randomly assigning one state a probability $\geq .8$ and assigning the remaining probabilities uniformly. To generate the incomplete structure G, we abstract all ancestor of query node Q by reducing their cardinality to 2. When parameterizing this structure to yield a TBN, we make every child of an abstracted node a testing node and use N evenly-spaced intervals for selecting a CPT.

Size	Handcrafted TBN	Learned TBN
10	0.0080	0.0383
15	0.0135	0.0248
20	0.0071	0.0172
25	0.0213	0.0117
30	0.0223	0.0220
35	0.0227	0.0107
40	0.0209	0.0091
45	0.0134	0.0018
50	0.0043	0.0039

Table 2. Conditional KL-divergence between handcrafted/learned TBNs and true BNs. Data points are averages over 50 trials.

Size	BN/AC	TBN/TAC	Best N	Gain (%)
10	0.0228	0.0141	4	38
20	0.0104	0.0068	8	35
30	0.0155	0.0078	8	50
40	0.0044	0.0034	2	23
50	0.0020	0.0013	4	35

Table 3. Conditional KL-divergence between learned BNs/TBNs and true BNs. Each data point is an average over 50 random trials.

Table 1 reports the conditional KL-divergence against the true BN. As we can see, the TBN yields much better recovery compared to both BN baselines and is close to the ground-truth BN when the number of intervals N is large enough. For $N = 2$, the TBN improves the KL-divergence by 10-20% compared to BN_SOFT. For $N = 50$, the TBN significantly improves the recovery quality by 50-90%. Figure 5 shows the quality of TBN recovery for up to $N = 100$.

6.2. Supervised Learning with Missing States

The setting for this task is similar to the previous task except that we now *learn* BN/TBN parameters from data using an incomplete structure G' (with abstracted nodes). First, we randomly generate polytree BNs and queries $Pr(Q|\mathbf{E})$ as discussed earlier. Next, we generate labeled datasets $\langle \mathbf{E}, Q \rangle$ from the BNs (true models), each including $100 \times n$ examples where n is the number of distinct evidence instantiations. Finally, we compile the incomplete structure G' and query into an AC (Arithmetic Circuit) and into a TAC (Testing Arithmetic Circuit) using the PyTAC system (Darwiche, 2020).[8] These circuits are computation graphs, in the form of tensor graphs, which compute the query based on a BN (AC) or a TBN (TAC), therefore facilitating the learning of their parameters using gradient descent.[9]

[8]The largest circuits we compiled had on the order of $10K$ nodes and their training took on the order of few minutes.

[9]To facilitate backpropagation, we use a sigmoid function instead of thresholds for placing the parent posterior into an interval.

True Model	Card	BN/AC	TBN/TAC	Best N	Gain(%)
2nd-HMM	2	0.1103	0.0406	4	63
	3	0.3028	0.1391	4	54
	4	0.5499	0.2502	8	54
3rd-HMM	2	0.2195	0.0490	8	77
	3	0.3676	0.1477	4	60
	4	0.6320	0.2005	8	68

Table 4. Conditional KL-divergence for learning a function using a first-order HMM when the data is generated by a high-order HMM. Each data point is an average over 30 random trials.

We conduct two experiments for this task. In the first, we compare learned TBNs with handcrafted ones. In the second, we compare learned TBNs with learned BNs. When learning TBN parameters, we also learn the interval boundaries instead of using equally-spaced intervals.

Table 2 reports the results of the first experiment for $N = 8$. When the network size is small, the learned TBN does not recover the query as well as the handcrafted one, but the learned TBN improves with increasing network size and dominates the handcrafted TBN with 25 or more nodes. Recall that for the handcrafted TBN, only children of abstracted nodes are testing so only these children have dynamic CPTs; all other nodes have fixed CPTs copied from the true BN. For the learned TBN, we make every node testing except for root nodes as we assume that we do not know the nature of modeling errors. As a result, the learned TBN has more degrees of freedom than the handcrafted one.

Table 3 depicts the results of the second experiment, where we use a number of intervals $N \in \{2, 4, 6, 8\}$ and report the best function $Pr'(Q|\mathbf{E})$ learned by a TBN. Our learning algorithm is not optimal or robust enough so the quality of learned functions was sometimes better for a smaller N. Yet, as shown by Table 3, a TBN with learned parameters holds the promise of recovering from modeling errors as in this case it did lead to 20-50% reduction in the conditional KL-divergence from the true function. That is, learning a set of distributions based on an incomplete structure (and then using one of them based on evidence) can allow one to learn better functions compared to learning a single distribution which is the case when learning a BN.

6.3. Supervised Learning with Missing Edges

We finally consider recovery from modeling errors in the form of missing edges. The true BN in this experiment is not a polytree, but becomes a polytree due to missing edges.

We assume the data is generated by an n-th order HMM (true model) but learn using a 1st-order HMM. We use HMMs with ten hidden variables $H_1, \dots, H_{10}$ and ten sensors $E_1, \dots, E_{10}$ and learn the function $Pr(H_{10}|E_1, \dots, E_9)$. We experiment with 2nd-order and 3rd-order HMMs as the true model, each with variables of cardinality 2-4 (six combinations). The dataset size was 512 for cardinality 2 and $20,000$ for cardinalities 3 and 4. We use a uniform initial distribution. For the emission distribution $Pr(E_t|h_t)$, we use a random $p \geq 0.95$ for $e_t = h_t$ and uniformly distribute $1 - p$ over $e_t \neq h_t$. For the transition distribution $Pr(H_t|h_{t-1}, \dots, h_{t-n+1})$, we pick a state h_t randomly and assign it a probability $p \geq 0.8$. We repeat this to assign a probability $\geq .8 * (1 - p)$ for a second state and so on.

Table 4 reports the quality of learned functions using a BN (static CPTs) and a TBN (dynamic CPTs). Again, we see that the TBN can help in recovering from this modeling error, as it did lead to gains up to 77% in the conditional KL-divergence in this case. (Shen et al., 2019) reported related experiments but using more restricted transition distributions. We obtain broader results as we use more than two CPTs per testing node. We now also have a theoretical basis for justifying the ability of TBNs to recover from such modeling errors.

7. Conclusion

We investigated the recovery from some modeling errors when learning probabilistic queries based on Bayesian network structures. These modeling errors induce independence assumptions that may not hold in the data-generating function, which contributes to bias. We cast these modeling errors in terms of state-space abstractions and provided conditions under which one can fully recover from these errors, assuming one can choose the CPTs of a Bayesian network dynamically based on evidence. We then showed how Testing Bayesian Networks can be used to realize the recovery proposal which revealed their promise as a tool for recovering from modeling errors. In future work, we plan to weaken the conditions under which full recovery can be attained, cast more modeling errors in terms of state-space abstractions and improve the learning of TBNs to further approach the promise of theoretical recovery results.

Acknowledgements

This work has been partially supported by grants from NSF IIS-1910317, ONR N00014-18-1-2561, and DARPA N66001-17-2-4032.

References

Bengio, Y., Lamblin, P., Popovici, D., and Larochelle, H. Greedy layer-wise training of deep networks. In *Advances in Neural Information Processing Systems 19 (NIPS)*, pp. 153–160, 2006.

Bishop, C. M. Model-based machine learning. *Philosophi-*

cal Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences, 371(1984):20120222, 2013.

Boutilier, C., Friedman, N., Goldszmidt, M., and Koller, D. Context-specific independence in bayesian networks. In *UAI*, pp. 115–123. Morgan Kaufmann, 1996.

Chang, K. and Fung, R. M. Refinement and coarsening of bayesian networks. In Bonissone, P. P., Henrion, M., Kanal, L. N., and Lemmer, J. F. (eds.), *UAI '90: Proceedings of the Sixth Annual Conference on Uncertainty in Artificial Intelligence, MIT, Cambridge, MA, USA, July 27-29, 1990*, pp. 435–446. Elsevier, 1990.

Chen, Y., Choi, A., and Darwiche, A. Supervised learning with background knowledge. In *International Conference on Probabilistic Graphical Models*, pp. 89–100. PMLR, 2020.

Choi, A. and Darwiche, A. On the relative expressiveness of bayesian and neural networks. In *PGM*, volume 72 of *Proceedings of Machine Learning Research*, pp. 157–168. PMLR, 2018.

Choi, A., Wang, R., and Darwiche, A. On the relative expressiveness of bayesian and neural networks. *International Journal of Approximate Reasoning*, 113:303–323, 2019.

Darwiche, A. A differential approach to inference in bayesian networks. *J. ACM*, 50(3):280–305, 2003.

Darwiche, A. *Modeling and Reasoning with Bayesian Networks*. Cambridge University Press, 2009.

Darwiche, A. Human-level intelligence or animal-like abilities? *Communications of the ACM*, 61(10):56–67, 2018.

Darwiche, A. An advance on variable elimination with applications to tensor-based computation. In *ECAI*, volume 325 of *Frontiers in Artificial Intelligence and Applications*, pp. 2559–2568. IOS Press, 2020.

Elidan, G. and Friedman, N. Learning the dimensionality of hidden variables. In Breese, J. S. and Koller, D. (eds.), *UAI '01: Proceedings of the 17th Conference in Uncertainty in Artificial Intelligence, University of Washington, Seattle, Washington, USA, August 2-5, 2001*, pp. 144–151. Morgan Kaufmann, 2001.

Elidan, G., Lotner, N., Friedman, N., and Koller, D. Discovering hidden variables: A structure-based approach. In *Advances in Neural Information Processing Systems*, pp. 479–485, 2001.

Gao, C., Sun, H., Wang, T., Tang, M., Bohnen, N. I., Müller, M. L., Herman, T., Giladi, N., Kalinin, A., Spino, C., et al. Model-based and model-free machine learning techniques for diagnostic prediction and classification of clinical outcomes in parkinson's disease. *Scientific reports*, 8(1):1–21, 2018.

Goodfellow, I., Bengio, Y., and Courville, A. *Deep Learning*. MIT Press, 2016.

He, Y. Extended viterbi algorithm for second order hidden markov process. In *9th International conference on pattern recognition*, pp. 718–719. IEEE Computer Society, 1988.

Hinton, G. E., Osindero, S., and Teh, Y. W. A fast learning algorithm for deep belief nets. *Neural Computation*, 18 (7):1527–1554, 2006.

Ide, J. S. and Cozman, F. G. Random generation of bayesian networks. In *Brazilian symposium on artificial intelligence*, pp. 366–376. Springer, 2002.

Kim, M. and Leskovec, J. The network completion problem: Inferring missing nodes and edges in networks. In *Proceedings of the 2011 SIAM International Conference on Data Mining*, pp. 47–58. SIAM, 2011.

Koller, D. and Friedman, N. *Probabilistic Graphical Models - Principles and Techniques*. MIT Press, 2009.

Murphy, K. P. *Machine learning - a probabilistic perspective*. MIT Press, 2012.

Pearl, J. *Probabilistic reasoning in intelligent systems - networks of plausible inference*. Morgan Kaufmann, 1989.

Ranzato, M., Poultney, C. S., Chopra, S., and LeCun, Y. Efficient learning of sparse representations with an energy-based model. In *Advances in Neural Information Processing Systems 19 (NIPS)*, pp. 1137–1144, 2006.

Shen, Y., Huang, H., Choi, A., and Darwiche, A. Conditional independence in testing bayesian networks. In *International Conference on Machine Learning*, pp. 5701–5709, 2019.

Wellman, M. P. and Liu, C.-L. State-space abstraction for anytime evaluation of probabilistic networks. In *Uncertainty Proceedings 1994*, pp. 567–574. Elsevier, 1994.