Bayesian Analysis (0000) **00**, Number 0, pp. 1

Learning to Anticipate Indirect Causes in Dynamic Bayesian Networks

Alexander Motzek[*,†], Ralf Möller[*]

Abstract. Modeling causal dependencies often demands cycles at a coarse-grained temporal scale. If Bayesian networks are used for representing uncertainty in temporal knowledge bases, cycles are frequently eliminated with dynamic Bayesian networks over time, which, however, spreads indirect dependencies over time as well, and enforces an infinitesimal resolution of time. Recently it has been shown that if indirect dependencies are spread over time, spurious results are returned and models can not represent causalities correctly. As a solution, it has been shown that some dynamic Bayesian networks, called ADBNs, are able to resolve cyclic dependencies intrinsically by a rapid adaptation to specific contexts at every timestep. To learn such networks from long but incomplete datastreams, parameter and structure learning of DBNs are fused into one atomic phase in this article. We show that classic (dynamic) Bayesian networks are unable to learn an anticipation of indirect causes in certain domains, and that classic approaches are not applicable to learning ADBNs. We proposes a learning approach for ADBNs considering rapidly adapting structures, while preserving (A)DBNs as a first-class representation of uncertainty in temporal knowledge bases.

Keywords: learning Bayesian networks, cyclic dependencies, indirect influences, context-specific independencies, activator dynamic Bayesian networks.

1 Introduction

Dynamic Bayesian networks (DBNs) are an extension to Bayesian networks motivated from two perspectives, on the one hand as a manifestation of cyclic dependencies over time, closely related to Markov models (Murphy, 2002), on the other hand as a stationary process repeated over time in fixed time slices (Glesner and Koller, 1995) for reasoning about historical and future evolutions of processes from uncertain, temporal knowledge bases. Considering Pearl and Russell (2003) who emphasized that Bayesian networks should be a direct representation of the world instead of a reasoning process, both views seem to be conflicting. A model repeated over time with cyclic dependencies would expand to infinity already for one timeslice. Therefore, e.g., Jaeger (2001) or Glesner and Koller (1995) use a strict order of dependencies s.t. state variables of time t are only dependent of states at $t-1$. However, Motzek and Möller (2015b) show that such non-causal models return spurious results for non-infinitesimal small timeslices, as indirect influences are not anticipated anymore.

In the most extreme form of a DBN, all random variables are seen as being locally dependent on each other, inevitably creating cyclic dependencies. In certain domains,

[*]Universität zu Lübeck, Institut für Informationssysteme, Germany. motzek@ifis.uni-luebeck.de moeller@uni-luebeck.de.
[†]Corresponding author

 DOI: 0000

dependencies depend on a specific context formed by further random variables as shown by Boutilier et al. (1996). Then, one can create a sound (acyclic) dynamic Bayesian network (DBN), once one knows the formed context for every timeslice, e.g., knows all future observations. However, as one can not know the future and a context changes at every timeslice, a required *structure can not be known in advance*. To include all possible dependencies in a classic DBN, these allegedly cyclic dependencies must be bend to a previous timestep, creating "diagonal" models. Motzek and Möller (2015b) show that a causal parametrization of such diagonal models is impossible and indirect influences are not anticipated. This raises an interesting question addressed in this article: *Can such "diagonal" models* learn *a set of parameters from data containing indirect influences, such that the learned model mimics the anticipation of indirect influences?* The short answer is: No, a learned diagonal model leads to completely different inference results.

Furthermore, Motzek and Möller (2015b) show that in such domains, DBNs are well-defined even for cyclic dependencies. Most importantly they show that these DBNs, called ADBNs, do not require external frameworks, remain in a classical calculus with classical random variables, and precisely anticipate the intended indirect influences based on a cyclic graph. It is shown that these ADBNs are able to rapidly adapt their structure *intrinsically* at every timestep to a specific context and, thus, correctly anticipate indirect influences. These contexts are specific instantiations of random variables in a timeslice. This raises another interesting question addressed in this article: *If these instantiations are missing in data for learning, a structure to which this ADBN adapts cannot be "seen" from data; does this still allow learning? Does one require heuristics to "guess" a structure?* The short answer, evaluated in this article is: By a modified EM algorithm, ADBNs learn parameters closely representing original models and deliver nearly identical inference results without a need for external frameworks "analyzing or guessing" a structure.

Different approaches for learning DBNs under an unknown structure exist, such as the structural EM algorithm by Friedman et al. (1998) and also for (slowly) changing structures over time, e.g., by Robinson and Hartemink (2010). Still, even if one would adapt an approach for a rapid structural change at every timestep, as required in ADBNs, existing approaches are not applicable. Learning is almost always separated into a threefold (incrementally repeated) problem of selecting/optimizing a structure, restoring values of missing values, and optimizing/learning new parameters. It is crucial to note that a structure is assumed to be constant for the act of restoring values of missing variables, i.e., a distribution of potential value-candidates is calculated based on *one* (previously optimized) structure. In ADBNs, a required structure is not knowable in advance and a required structure is only identifiable in retrospect given a specific context of each timeslice. This means, there exists no single structure on which value candidates can be inferred. If inference is performed on one single structure, some missing values of a datapoint (that shall be inferred) are fixed immediately by the choice of one single structure. Effectively, there exists no difference between structure- and parameter-learning in ADBNs.

This paper can be summarized as follows: We show that classic DBN formalisms are neither able to anticipate nor are they able to learn indirect influences in certain domains.

Representing multiple structures in one ADBN model anticipates indirect influences correctly, and fuses structure- and parameter learning into one atomic task. We propose a learning approach for ADBNs for problems where effective structures are not (and need not to be) known in advance, are rapidly changing over time, and are not evident from data while learning. This is beneficial for applications where the interpretation of data depends on further contexts and is otherwise neither explicable nor learnable, e.g., in complex probabilistic, temporal knowledge bases.

We motivate and introduce ADBNs in Section 2 by a running example, and derive a learning approach for ADBNs for incomplete datasets in Section 3. Based on an EM approach, we show that classic DBNs are unable to learn (from) indirect influences and experimentally evaluate different scenarios of hidden variables in ADBNs in Section 4. We discuss our results and related work in Section 5 and conclude with Section 6.

2 Activator Dynamic Bayesian Networks

After initial notations and definitions used throughout this paper, we demonstrate a running example for ADBNs. We consider an example based on Motzek and Möller (2015b) which outlines the necessity of ADBNs and shows why classic Bayesian networks and associated learning approaches are not applicable in certain domains. We consider DBNs describing a probabilistic process over time, where time represents a causal flow of wallclock-time at a specific granularity suited to a given process. A common structure over time allows for reasoning about an evolution of values of random variables, which are defined as follows.

Notation 1 (State Variables)**.** *Let X_i^t be the random variable representing the i^{th} state variable X_i at time t, where X_i^t is assignable to a value $x_i \in \mathrm{dom}(X_i^t)$. Let $\vec{X}^t$ be the vector of all n state variables at time t, s.t.,*

$$\vec{X}^t = \left(X_1^t, \ldots, X_n^t\right)^\intercal .$$

A random variable is a state variable if it bears a history, i.e., is at least dependent on X_i^{t-1}. Let $P(X_i^t = x_i)$ (or $P(x_i^t)$ for brevity) denote the probability of state X_i having x_i as a value at time t. If $\mathrm{dom}(X) = \{true, false\}$ we write $^+x^t$ for the event $X^t = true$ and $\neg x^t$ for $X^t = false$ as usual. If X_i^t is unspecified and not fixed by evidence, $P(X_i^t)$ denotes the probability distribution of X_i^t w.r.t. all possible values in $\mathrm{dom}(X_i^t)$. ▲

As usual, dependencies between random variables of and inside consecutive timeslices exist, forming a dynamic Bayesian network (DBN) allowing for reasoning about evolutions of values of random variables over time.

Definition 1 (Dynamic Bayesian Network)**.** *A DBN is a tuple $(B_0, B_\rightarrow)$ with B_0 defining an initial Bayesian network (BN) representing time $t = 0$, containing all state variables X_i^0 in $\vec{X}^0$, and a consecutively repeated Bayesian network fragment $B_\rightarrow$ defining state dependencies between X_i^s and X_j^t, with $X_i^s \in \vec{X}^s, X_j^t \in \vec{X}^t, s \leq t$. For every random variable X_i^t a local conditional probability distribution (CPD) $P(X_i^t | parents(X_i^t))$, e.g., as a CPT, is defined, where $parents(X_i^t)$ is the set of parents of X_i^t in B_0 or, respectively,*

$B_{\rightarrow}$. By repeating $B_{\rightarrow}$ for every time step $t > 0$, a DBN $(B_0, B_{\rightarrow})$ is unfolded into a Bayesian network defining its semantics as the joint probability over all random variables $P(\vec{X}^{0:t^\intercal})$ as the product of all locally defined CPDs.

Dependencies, i.e., edges, between random variables defined in B_0 and $B_{\rightarrow}$ are limited, s.t. no cyclic dependencies are created during unfolding. ▲

Motzek and Möller (2015b) introduce a running example that highlights an imminent problem of causally correctly representing a domain in a classic DBN formalism: a domain where some dependencies depend on specific instantiations of some random variables, which change at every time. To grasp all possible situations, a DBN must be designed proactively with cyclic dependencies, where an actual structure of each timeslice is first known once specific observations (or constraints) are known in the current timeslice, i.e., a required structure rapidly changes and adapts to a context formed at every timestep.

Example 1 (Taintedness domain)**.** *In a company, one is concerned with regulatory compliance of employees. A corrupt employee possibly manipulates and transfers documents, based on which a recipient might commit a compliance violation, too. Recipients unknowingly become corrupt as well, and spread false information throughout a company as well. As recipients become corrupt unknowingly, we speak of* taintedness *and that an employee is* tainted *in order to prevent a sense of false accusation. To trace back a potential source for compliance violations and to reconstruct sequences of events, we model a probabilistic regulatory-compliance domain using a DBN. The local interpretability of specified conditional probability distributions allows for an intuitive design and, when learned, deliver valuable information for themselves. We represent the taintedness of an employee, say, C^tlaire, D^ton, E^tarl by state random variables in $\vec{X}^t$. An employee X influences another employee Y by a sent message at time t, represented by random variable M^t_{XY}. Then, $P(+c^1|+m^1_{DC})$ represents the probability that Claire is tainted at time 1, given that we know that (at least) a message was transferred from Don to Claire at time 1. Say, one collects message transfer observations from used office mail envelopes that have been used during one day. Usually, such envelopes are reused multiple times and one finds multiple transfers in an unknown temporal ordering for each day.*

We intend to model that every employee is able to send a message to every other employee. *This means, locally seen, every employee, i.e., state variable, is dependent on every other, which inevitably creates cyclic dependencies as shown in Figure 1, which we call* cyclic. ♦

The example shows that cyclic dependencies arise naturally in domains and are required from a causal perspective. Note that employees, as well as message transfers are classical random variables for which plain CPDs are specified. However, to maintain in a classic DBN formalism, the only consequent option keeping the same number of parameters and dependencies is to bend all dependencies to a consecutive timestep (see Fig. 1, gray), creating a *diagonal* structure. However, as shown by Motzek and Möller (2015b) and outlined in Example 2, indirect influences are then not anticipated anymore during one timeslice. If Claire sends a message to Don and Don sends a message to Earl, no indirect influences of Claire on Earl is anticipated anymore in a diagonal model. In

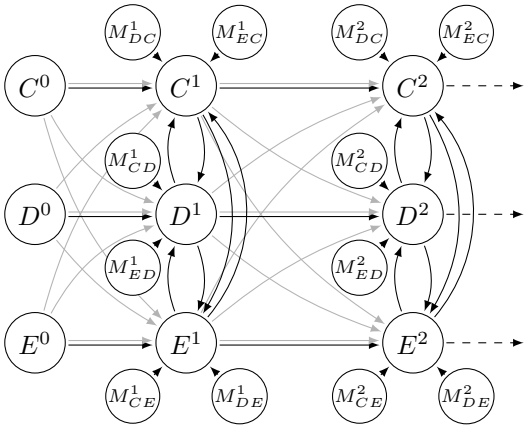

Figure 1: Two options (black/gray) to represent the taintedness using an (A)DBN where every employee (*C*laire, *D*on, *E*arl) can influence every other employee through messages M^t_{XY}. Syntactic DAG constraints of (D)BNs prevent cyclic dependencies and diagonal (indicated with gray arrows) state dependencies are enforced. A cyclic ADBN (black) is well-defined under similar DBN semantics and represents a world more accurately than the diagonal option does.

fact, a diagonal option models a different domain, where an incubation period of one timeslice (e.g., 1 day) is modeled.

In fact, a cyclic taintedness domain is directly evident in a cyber security application, assessing how locally inflicted adverse effects on devices or services (taintedness of employees) "spread" throughout a network by datatransfers (message transfers) leading to causal chain of events potentially striking highly critical devices. If highly critical devices are potentially not operating as intended, a complete company might fail their business goals, or missions are not accomplishable anymore. Understanding the indirect and transitive effects is frequently called a *mission impact assessment*. Motzek and Möller (2016) discuss an extension of Motzek et al. (2015) for a probabilistically sound mission impact assessments extended towards dynamic domains based on ADBNs for ongoing and retrospective analysis in dynamic mission impact assessments. Due to locally interpretable and understandable parameters, (A)DBNs are predestined for such an application, and it is highly desirable to learn corresponding ADBNs, in order to correctly obtain models for adverse affections through compromised data-transfers.

Fortunately, Motzek and Möller (2015b) show that the cyclic option outlined in Example 1 is indeed a well-defined dynamic probabilistic graphical model and is remarkable similar to a DBN, as message transfer variables represent, so called, activator random variables. Activator random variables represent random variables for which CPDs of state variables show certain properties. We use the notation A_{XY} if a random variable acts as an activator random variable which activates a dependency of random variable Y on X in a given context.

Definition 2 (Activation / deactivation criteria)**.** *Let* $\mathrm{dom}(A_{XY}) = \{true, false\}$. *Motzek and Möller (2015b) define the* deactivation *criterion* $A_{XY} = false$ *as*

$$\forall x, x' \in \mathrm{dom}(X), \forall y \in \mathrm{dom}(Y), \forall \vec{z} \in \mathrm{dom}(\vec{Z}) :$$
$$P(y|x, \neg a_{XY}, \vec{z}) = P(y|x', \neg a_{XY}, \vec{z}) = P(y|*, \neg a_{XY}, \vec{z}) \ ,$$

where $*$ *represents a wildcard and* $\vec{Z}$ *further dependencies of* Y*. Given* $\neg a_{XY}$*, ones says that the dependence of* X *on* Y *is* inactive*. The* activation *criterion describes a situation where* Y *becomes dependent on* X*, i.e., the CPD entry for* y *is not uniquely identified by just* $+a_{XY}$ *and* $\vec{z}$*, hence*

$$\exists x, x' \in \mathrm{dom}(X), \exists y \in \mathrm{dom}(Y), \exists \vec{z} \in \mathrm{dom}(\vec{Z}) :$$
$$P(y|x, +a_{XY}, \vec{z}) \neq P(y|x', +a_{XY}, \vec{z}) \ .$$

Given $+a_{XY}$*, one says that the dependence of* X *on* Y *is* active*.*

If for a random variable A^t_{XY} *both activation and deactivation criteria are fulfilled by a CPD definition of random variable* Y*,* A^t_{XY} *is called an* activator *random variable. Then, let* A^{st}_{ij} *be an activator random variable regarding a dependency from* X^t_j *on* X^s_i*. Let* $\vec{\mathcal{A}}^{s\,t}$ *represent the vector of all activator random variables between timeslices* s *and* t*. For brevity, we write* A^t *for* A^{tt}*, and let* $\vec{A}^t_i$ *represent the vector of all activators relevant for* X^t_i*, i.e., all* A^{*t}_{*i}*.* ▲

In fact, the running example is a DBN in which a subset of random variables shows to have activator nature according to Definition 2. In detail, random variables representing message exchanges in Example 1 are activator random variables and, therefore, Example 1 represents an ADBN:

Definition 3 (Activator dynamic Bayesian network, ADBN)**.** *An ADBN is syntactically defined as a tuple* $(B_0, B_\rightarrow)$ *with* B_0 *defining an initial Bayesian network (BN) representing time* $t = 0$ *containing all states* $\vec{X}^0_i \in \vec{X}^0$ *and a consecutively repeated activator BN fragment* $B_\rightarrow$ *consisting of dependencies between state variables* X^s_i *and* X^t_j*,* $t-1 \leq s \leq t$ *(Markov-1), and dependencies between state variables* X^t_i *and activator random variables* A^{st}_{ji}*. For every random variable* X^t_i*,* A^{st}_{ij} *a local CPD over all parents, e.g., as a CPT, is specified, where CPDs of state variables* X^t_i *follow Definition 2 s.t. random variables* A^{st}_{ij} *are activator random variables.*

By repeating $B_\rightarrow$ *for every time step* $t > 0$*, an ADBN* $(B_0, B_\rightarrow)$ *is unfolded into a BN defining ADBN semantics as the joint probability over all random variables* $P(\vec{X}^{0:t\intercal}, \vec{\mathcal{A}}^{01:tt\intercal})$*.* ▲

Please note that activators in an ADBN are classic random variables and are part of a modeled domain, i.e., activators are no auxiliary variables. ADBNs are restricted in order to comply with a BN, i.e., follow a different well-definedness theorem:

Theorem 1 (ADBN well-definedness)**.** *An ADBN is well-defined, if it is a well-defined DBN. An ADBN is well-defined for every* regular *instantiation* $\vec{a}^{1:t}$ *of* $\vec{\mathcal{A}}^{1:t}$*, i.e., if for all* t*,* $\vec{a}^t$ *satisfies the acyclicity predicate* $\mathfrak{A}$*:*

$$\forall x, y, z \in \vec{X}^t : \mathfrak{A}(x,z)^t, \mathfrak{A}(z,y)^t \rightarrow \mathfrak{A}(x,y)^t$$
$$\neg \exists q : \mathfrak{A}(q,q)^t \ ,$$

with a acyclicity predicate $\mathfrak{A}(i,j)^t$ *that is defined as*

$$\mathfrak{A}(i,j)^t = \begin{cases} \mathit{false} & \mathit{if} \quad \neg a_{ij}^t \in \vec{a}^t \\ \mathit{true} & \mathit{if} \quad \mathit{else} \end{cases} .$$

For every well-defined ADBN, the semantics as $P(\vec{X}^{0:t^\intercal}, \vec{\mathcal{A}}^{01:tt^\intercal})$ *is sound and equivalent to DBN semantics as the product of all locally defined CPDs.* ▲

A proof for this theorem is given by Motzek and Möller (2015b, Sec. 3). Theorem 1 means that ADBNs are subject to a different acyclicity constraint, namely that not a syntactic graph structure is forced to be acyclic, but rather that an instantiation of an unrolled DBN becomes acyclic in each timestep. This means, observations or parameter setting properties have to enforce that only well-defined *regular* instantiations are used for inference. Informally Theorem 1 means that, if one could known all future observations in advance, one could create a well-defined acyclic BN for a specific application proactively. However, required structures are not knowable in advance and are changing at every timestep depending on a specific context, i.e., an instantiation of some activator random variables.

Please note that in ADBNs no external frameworks are involved analyzing a required structure from observations, and that the ADBN formalism intrinsically handles the rapid structural change that is required at every timestep without any introduction of a novel calculus. In essence, an ADBN represents a classical DBN with classical random variables, where some specific number constellations in CPDs allow for cyclic dependencies. On top of that, Motzek and Möller (2015b) shows that solutions to filtering and smoothing problems in ADBNs remain in the same complexity class as multiply connected DBNs and, further, show that classical algorithms such as the forward-backward algorithm remain applicable. Informally it can be said that Motzek and Möller (2015b) show that DBNs are able and have been able ever since to be based on cyclic graphs for specific number constellations in CPDs, despite every introduction to classical Bayesian networks who almost always enforce DAGs. The following example further motivates the desire to preserve context-specific cyclic dependencies in the running example.

Example 2 (ADBN well-definedness example)**.** *Continuing the running example, suppose, one observes that Don and Earl were initially not corrupt* $\neg d^0, \neg e^0$*, but Earl is convicted at time* 1 *of compliance violations, i.e., one observes* $+e^1$*. Further, one observes transfers* $\neg m_{DC}^1, \neg m_{ED}^1, \neg m_{EC}^1, \neg m_{CE}^1$*. Then, the cyclic option is a well-defined ADBN and anticipates a (potential) indirect influence of Claire on Earl, explaining the source of Earl's violation by* $+c^0, +m_{CD}^1, +m_{DE}^1$*. One can say that, an ADBN rapidly adapts its structure to the made observations and Claire influences Earl indirectly during day* 1*.*

However, in a diagonal option, no influence is exerted by Claire on Earl, and an obtained inference result would be spurious, as Earl becoming tainted is inexplicable: The only explanations for Earl's violation are being corrupt in the first place ($+e^0$), or becoming tainted through Don's message ($+d^0, +m^1_{DE}$) which are both contrary to the made observations. The problem is that an indirect influence is spread over multiple timeslices, i.e., over multiple days, e.g., day-0-Claire can first influence day-2-Earl in a diagonal option. ♦

The example shows that cyclic dependencies are required in order to be a declarative first-class world representation. Motzek and Möller (2015b) conclude that diagonal (A)DBNs can not anticipate indirect causes, which enforces an infinitesimal resolution of time, e.g., seconds, where indirect influences do not *need* to be considered. Moreover, this example shows that diagonal (A)DBNs cannot be parametrized causally, i.e., from local perspectives in pure cause→effect-relationships: the introduced "incubation-"period must somewhat be overcome by spurious modifications to CPDs mimicking an anticipation of indirect influences.

The running example has been especially chosen, as it represents a superclass of all potential intra-timeslice DBNs (called a dense intra-timeslice structure). Note that its defined joint probability distribution (JPD) is remarkably similar to a classic DBN approach:

Proposition 1 (ADBN semantics)**.** *If an ADBN is well-defined (Thm. 1), the JPD over all random variables is defined as the product of all locally defined CPDs. Therefore, a dense intra-timeslice ADBN's semantics is*

$$P(\vec{X}^{0:t^\intercal}, \vec{\mathcal{A}}^{1:t^\intercal}) = \prod_{X^0_k \in \vec{X}^0} P(X^0_k) \cdot \prod_{i=1}^{t} \prod_{A^i_{cv} \in \vec{\mathcal{A}}^i} P(A^i_{cv}) \cdot \prod_{X^i_k \in \vec{X}^i} P(X^i_k | \vec{X}^{i^\intercal} \setminus X^i_k, \vec{A}^{i^\intercal}_k, X^{i-1}_k) \ . \quad (2.1)$$

▲

This proposition shows that an ADBN based on a cyclic graph still remains in a similar calculus as classical DBNs, i.e., no novel operators are introduced and CPDs of random variables are still handled as one is used to. In effect, this JPD encompasses all potential intra-timeslice (A)DBNs and is therefore further used in derivations by which one obtains universally applicable results.

In summary: In an ADBN, a specific context of instantiations (activators) steers an ADBN's structure. This means, model parameters and instantiations influence a structure as well as an initially defined general structure. Given, say, a long but incomplete period of observations of compliance infringements in a company, it is of utmost interest to learn a cyclic ADBN to precisely predict future compliance violations and to reconstruct events that lead to previous infringements.

Intending to learn ADBNs now raises two interesting questions: If in ADBNs a structure is not known in advance, changes at every timestep, and is firstly known "live" once a timeslice is (partially) instantiated, can we still learn ADBNs and can we still learn from incomplete datasets? Do we require an external analysis of a structure at every timestep?

A first thought could be to utilize learning approaches for changing structures over time, e.g., by Robinson and Hartemink (2010) (cf. Section 5). However, such learning approaches do often neither consider hidden variables nor incomplete datasets and only consider slowly changing structures that cannot keep the pace of changing structures, which is required in ADBNs. Moreover, even modifications of such learning approaches to support rapidly changing structures under hidden variables can not learn ADBNs, as there exists a more imminent problem: If an ADBN's effective structure depends on an instantiation, there exists no fixed structure for one timeslice on which a partially unobserved instantiation can be restored. If one fixes a structure prior to inference per timeslice, instantiations that shall be inferred, are fixed as well. Namely, every missing edge in a structure, fixes the instantiation of associated activators to be deactive. Therefore, inference must be performed on all possible structural candidates simultaneously. Effectively, there exists no difference between parameter and structural learning in ADBNs. In order to learn ADBNs, a different approach must be taken, and in the following we show that a variation of an EM algorithm is able to learn cyclic ADBNs from incomplete data under partially missing structural information, *without* an explicit analysis of a structure of each timeslice and without introducing an overhead, compared to classical approaches.

3 Learning Structures where Structures are not Knowable

Learning is based on large amounts of data representing a (partially) observed state of a process. Therefore, a dataset $\vec{d}^{0:t}$ is a time series of instantiations of some random variables in $\vec{X}^{0:t}$ and $\vec{\mathcal{A}}^{1:t}$. Without loss of generality, we consider that learning is performed only on a single (long enough) time series $\vec{d}^{0:t}$ of data. This means, a DBN represents a process evolving over time, where time represents an actual irreversible flow of time at a specific granularity, instead of an arbitrary construct of operation-"time"-slots. This means, an initial BN fragment B_0 is only instantiated once, making it impossible to learn prior probabilities of B_0.

Often, learning-approaches and, especially, EM-approaches are introduced informally, by reducing learning to a simpler case of complete datasets, i.e., all values of variables are available for learning. Learning parameters from complete datasets effectively reduces to counting how often an event associated with a parameter is seen, i.e., learning from complete datasets is linear in the number of parameters and the size of the dataset. Based on learning from complete datasets, learning from incomplete data is often introduced via an idea that virtual data counts are created by inferring all possible value candidates from one incomplete datum. One has to hopefully expect that such an approach remains applicable to cyclic ADBNs and that no external frameworks analyzing structures separately are required, but as multiple structures are represented by one cyclic ADBN and are subject to a novel acyclicity constraint, applicability is not granted. We therefore, carefully derive a classic EM approach from a pure probabilistic point of view throughout this section and evaluate its effectiveness in the upcoming section. To do so, one has to carefully differentiate between to-be-learned parameters and a numerical instantiation of them.

Notation 2 (Parameters)**.** *Let $\vec{\Theta}$ represent the vector of all to-be-learned parameters as variables. Let $\vec{\vartheta}$ represent a specific numerical instantiation of all parameters $\vec{\Theta}$. Let $P_{\vec{\vartheta}}(\cdot)$ represent a probability derived based upon a set of a parameter instantiation $\vec{\vartheta}$, i.e., $P_{\vec{\vartheta}}(\cdot)$ is a* number*. Let $P_{\vec{\Theta}}(\cdot)$ represent a probability based on a set of parameter variables in $\vec{\Theta}$, i.e., $P_{\vec{\Theta}}(\cdot)$ is an* equation *with variables. Given a set of instantiated parameters $\vec{\vartheta}$, there exists a specific probability $P_{\vec{\vartheta}}(\vec{d})$ of observing a dataset $\vec{d}$, i.e., the likelihood of the data.* ▲

A dataset $\vec{d}^{0:t}$ is seen as a probability distribution over all possible full-instantiations $\vec{x}^{0:t}$, $\vec{a}^{1:t}$ conforming with $\vec{d}^{0:t}$ under parameters $\vec{\vartheta}$ that could have been observed, also called the dataset distribution:

$$P_{\vec{\vartheta}}(\vec{X}^{0:t^\intercal}, \vec{\mathcal{A}}^{1:t^\intercal} | \vec{d}^{0:t}) = \alpha \cdot P_{\vec{\vartheta}}(\vec{X}^{0:t^\intercal}, \vec{\mathcal{A}}^{1:t^\intercal})$$

For inference, a dataset $\vec{d}^{0:t}$ is seen as observed random variables $(\vec{z}, \vec{b})^{0:t}$.

Notation 3 (Vectors of observed and unobserved variables)**.** *Let $\vec{Z}^t \subseteq \vec{X}^t$ be a vector of observed and $\vec{\zeta}^t = \vec{X}^t \backslash \vec{Z}^t$ be the vector of unobserved state variables. Let $\vec{\mathcal{B}}^t \subseteq \vec{\mathcal{A}}^t$ be a vector of observed activators. Likewise, let $\vec{\beta}^t = \vec{\mathcal{A}}^t \backslash \vec{\mathcal{B}}^t$ be a vector of all unobserved activators. Then, data $\vec{d}^t$ is seen as observations of state variables $\vec{z}^t$ (instantiation assignments $X_i^t = x_i \in \mathrm{dom}(X_i^t)$), and as observations of activator random variables $\vec{b}^t$ (instantiation assignments $A_{ij}^t = a_{ij} \in \mathrm{dom}(A_{ij}^t)$).* ▲

Given a dataset distribution $P_{\vec{\vartheta}}(\vec{X}^{0:t^\intercal}, \vec{\mathcal{A}}^{1:t^\intercal} | \vec{d}^{0:t})$, the probability of observing (note, not to have observed) $\vec{d}^{0:t}$ under parameters $\vec{\Theta}$ is

$$P_{\vec{\Theta}}(\vec{d}^{0:t}) = \sum_{\vec{\zeta}^{0:t}} \sum_{\vec{\beta}^{1:t}} P_{\vec{\vartheta}}(\vec{X}^{0:t^\intercal}, \vec{\mathcal{A}}^{1:t^\intercal} | \vec{d}^{0:t}) \cdot P_{\vec{\Theta}}(\vec{X}^{0:t^\intercal}, \vec{\mathcal{A}}^{1:t^\intercal}) \ , \tag{3.1}$$

further called the likelihood of the dataset, and where instantiations in $\vec{d}$ uniquely identify the instantiation of respective random variables in $\vec{D}$.

Following a maximum likelihood approach, it is the goal to find a parameter instantiation $\vec{\vartheta}^*$ of $\vec{\Theta}$ that maximizes the probability of observing $\vec{d}^{0:t}$, i.e.,

$$\vec{\vartheta}^* = \arg\max_{\vec{\Theta}} P_{\vec{\Theta}}(\vec{d}) = \arg\max_{\vec{\Theta}} \log\left(P_{\vec{\Theta}}(\vec{d})\right) \ , \tag{3.2}$$

where an expectation maximization (EM) approach iterates between calculating a distribution $P_{\vec{\vartheta}}(\vec{X}^{0:t^\intercal}, \vec{\mathcal{A}}^{1:t^\intercal} | \vec{d}^{0:t})$ (E-step) and Eq. 3.2 (M-step). In fact, $P_{\vec{\vartheta}}(\vec{X}^{0:t^\intercal}, \vec{\mathcal{A}}^{1:t^\intercal} | \vec{d}^{0:t})$ is calculated trivially, but requires an intractable amount of memory, which is why in the following an extended smoothing distribution is used.

Definition 4 (Extended smoothing problem)**.** *Given a probabilistic knowledge base B_0, $B_\rightarrow$, the extended smoothing problem is the problem of determining all extended smoothing distributions k, $k < t$ over all random variables in timeslices k and $k-1$ while considering evidence $\vec{z}^{0:t^\intercal}, \vec{\mathcal{b}}^{1:t^\intercal}$ until time t. This is, to obtain*

$$P(\vec{X}^{j^\intercal}, \vec{\mathcal{A}}^{j^\intercal}, \vec{X}^{j-1^\intercal}, \vec{\mathcal{A}}^{j-1^\intercal} | \vec{z}^{0:t^\intercal}, \vec{\mathcal{b}}^{1:t^\intercal}) \ , \ \forall j : 1 \leq j < t \ .$$

We denote a parametrized extended smoothing problem as $\texttt{ExtdSP}(B_0, B_\rightarrow, \vec{z}^{0:t}, \vec{\mathcal{b}}^{1:t}, t)$. ▲

Finding exact solutions to extended smoothing problems is discussed in the digital supplement, and, in summary, is linear in t like similar smoothing operations in classical DBNs and only requires storage for a distribution over all random variables of two timeslices.

For brevity, further derivations are given in the digital supplement in addition to the proof for the following theorem on the derived learning approach. Notwithstanding, when learning ADBNs one needs to consider activator criteria, for which we define the following.

Definition 5. *Let $X^i_\lambda \in \vec{X}^i$ be a state variable and let $P_\Theta(x^i_\lambda | \vec{x}^{i^\intercal} \backslash x^i_\lambda, \vec{a}^{i^\intercal}_\lambda, x^{i-1}_\lambda)$ be a to be learned parameter. We partition $\vec{a}^i_\lambda$ into two vectors $\vec{a}^i_\lambda = \langle {}^+\vec{a}^i_\lambda, \neg\vec{a}^i_\lambda \rangle$ containing active and inactivate activator random variables. Then, let $\vec{x}^{i^\intercal} \backslash x^i_\lambda = \langle \vec{x}^i_{\blacksquare\lambda}, \vec{x}^i_{\square\lambda} \rangle$ be a partition of relevant and irrelevant dependencies of X^i_λ under an instantiation $\vec{a}^i_\lambda$ respective to the activator criteria from Definition 2, such that*

$$\forall k \ {}^+a^i_{k\lambda} \in {}^+\vec{a}^i_\lambda : x^i_k \in \vec{x}^i_{\blacksquare\lambda}$$
$$\forall k \ \neg a^i_{k\lambda} \in \neg\vec{a}^i_\lambda : x^i_k \in \vec{x}^i_{\square\lambda} \ .$$

Respectively let $\vec{X}^i_{\blacksquare\lambda}, \vec{X}^i_{\square\lambda}$ represent the respective random variables of instantiations $\vec{x}^i_{\blacksquare\lambda}, \vec{x}^i_{\square\lambda}$. ▲

Using Definition 4 and Definition 5 one obtains an EM procedure.

Theorem 2 (EM procedure)**.** *Repeatedly evaluating*

$$P^*_{\vec{\vartheta}}(x^i_\lambda | \vec{x}^{i^\intercal} \backslash x^i_\lambda, \vec{a}^{i^\intercal}_\lambda, x^{i-1}_\lambda) = \frac{\gamma'(X_\lambda = x_\lambda)}{\gamma'(X_\lambda = {}^+x_\lambda) + \gamma'(X_\lambda = \neg x_\lambda)} \tag{3.3}$$

with
$$\gamma'(X_\lambda = x_\lambda) = \sum_{i=1}^{t} \sum_{\vec{\zeta}^{i-1} \backslash X^{i-1}_\lambda} \sum_{\vec{\beta}^{i-1}} \sum_{\vec{\beta}^i \backslash \vec{A}^i_\lambda} \sum_{\vec{X}^i_{\square\lambda}}$$
$$P_{\vec{\vartheta}}(\vec{X}^{i-1^\intercal} \backslash X^{i-1}_\lambda, x^{i-1}_\lambda, \vec{\mathcal{A}}^{i-1^\intercal}, \vec{X}^{i^\intercal}_{\square\lambda}, \vec{x}^{i^\intercal}_{\blacksquare\lambda}, \vec{a}^{i^\intercal}_\lambda, \vec{\mathcal{A}}^{i^\intercal} \backslash \vec{A}^i_\lambda | \vec{d}^{0:t}) \ ,$$

and

$$P^*_{\vec{\vartheta}}(A^i_{\mu\nu}) = \frac{\gamma'(A_{\mu\nu} = a_{\mu\nu})}{\gamma'(A_{\mu\nu} = {}^+a_{\mu\nu}) + \gamma'(A_{\mu\nu} = \neg a_{\mu\nu})} \tag{3.4}$$

with $\gamma'(A_{\mu\nu} = a_{\mu\nu}) = \sum_{i=1}^{t} \sum_{\vec{\zeta}^{i-1:i}} \sum_{\vec{\beta}^{i-1}} \sum_{\vec{\beta}^i \backslash A^i_{\mu\nu}} P_{\vec{\vartheta}}(\vec{X}^{i-1^\intercal}, \vec{\mathcal{A}}^{i-1^\intercal}, \vec{X}^{i^\intercal}, \vec{\mathcal{A}}^{i^\intercal} \backslash A^i_{\mu\nu}, a^{i^\intercal}_{\mu\nu} | \vec{d}^{0:t}) \ .$

for all parameters in $\vec{\Theta}$, is an algorithm that learns model parameters from incomplete data, where even structural information, i.e., activators, are hidden. Every iteration increases the likelihood of being able to observe $\vec{d}^{0:t}$ under an optimized parameter set $\vec{\vartheta}^*$ and converges to a local optimum. At every iteration, one evaluates $\gamma'(\cdot)$, and respectively solves $\texttt{ExtdSP}(B_0, B_{\rightarrow}, \vec{z}^{0:t}, \vec{b}^{1:t}, t)$ *to obtain* $P_{\vec{\vartheta}}(\vec{X}^{0:t\intercal}, \vec{\mathcal{A}}^{1:t\intercal} | \vec{d}^{0:t})$*, (*E*xpectation-step), and consecutively evaluates* $P^*_{\vec{\vartheta}}(\cdot)$*, (*M*aximization-step).* ▲

That the proposed algorithm maximizes the likelihood in each iteration is proven in the digital supplement by a continued, detailed derivation on the procedure, which can be summarized as follows: By summing out a specific parameter, Eq. 3.2, i.e., the maximization of the likelihood, is analytically solved in a closed form by partial derivation and finding zeros. Under careful consideration of identical parameters due to activator constraints, one obtains all equations stated by Theorem 2 after several transformations.

To summarize, a learning approach for dense intra-timeslice ADBNs, where a structure cannot be known in advance, shows similar structure to EM algorithms for DBNs. In fact, one obtains a commonly known form of "expected counts" (see also Russell and Norvig, 2010) as a closed form for optimized parameters $\vec{\vartheta}^*$, as hoped for. Note that, at no time an effective structure is made explicit, and throughout the learning procedure a structure of each timeslice remains unknown. Therefore, we say that learning ADBNs fuses structure- and parameter learning into one atomic phase.

In the previous section we have introduced the taintedness domain, where cyclic ADBNs arise naturally, and diagonal (A)DBNs cannot be parametrized causally. In the following section we learn taintedness domains from datasets and evaluate the proposed learning approach on cyclic ADBNs and answer the question, whether diagonal (A)DBNs could actually learn the taintedness domain s.t. diagonal models return similar inference results to the novel cyclic models.

4 Convergence and Hidden Variables

This section explores and empirically evaluates different situations of hidden variables while learning ADBNs. For empirical evaluation, we learn models from multiple datasets gathered from simulated taintedness domains over long periods of time with $N = |\vec{X}^t|$ employees, i.e., a long, but incomplete period of observations of compliance infringements in a virtual company. We generate these datasets by using particle filters under randomly generated CPDs following Definition 2 in the taintedness domain. To avoid impossible situations we constrain local CPDs to contain only probabilities in the range $[0.01, 0.99]$ and restrain all priors to be 0.5. As mentioned after Example 1, the taintedness domain reflects a cyber security application for a dynamic mission impact assessment, where, informally, infections or transitive effects of adversary actions "spread" throughout a network based on transferred data. Likewise, an application for the presented learning approach evaluated in this section is directly evident: datasets, from which such dynamic-mission-impact-assessments-ADBNs can be learned, are directly obtained by captured network traffic metadata, e.g., ".pcap-"files, and corresponding information about local impacts, e.g., intrusion protection and detection system, anti-virus scans, firewall breaches or retrospective manual analyses.

To prove the convergence of learned parameters $\vec{\vartheta}^*$ (as stated by Thm. 2) towards the original model parameters, we consider the absolute error between learned and original CPDs in different situations of missing data for cyclic and diagonal (A)DBNs. To validate learned models, we generate new sequences of observations and compare the accuracy of inference results for filtering and smoothing problems. To do so we calculate the Hellinger distance (a bound difference metric between $[0, 1]$, cf. Harsha and Varma, 2011) between results obtained by the learned model and results obtained from the original model. One needs to expect small differences of inference results, i.e., low, near 0 Hellinger distances.

In summary, more than 200 experiments confirm that learned CPDs, using the learning procedure from Theorem 2, converge to a local optimum that closely represents the original CPDs, even from incomplete datasets with a very low observability space (see Fig. 4.) In combination with the analytical proof of Theorem 2 in the digital supplement, we consider Theorem 2 as proven. □

In the following sections, we apply and discuss results of the learning procedure for different observability spaces and structural models.

4.1 Learning from complete data

If a dataset $\vec{d}$ contains only full instantiations $\vec{x}^{0:t}, \vec{a}^{1:t}$ of all variables $\vec{X}^{0:t^\intercal}, \vec{\mathcal{A}}^{1:t^\intercal}$, then a probability distribution $P_{\vec{\vartheta}}(\vec{X}^{0:t^\intercal}, \vec{\mathcal{A}}^{1:t^\intercal} | \vec{d}^{0:t})$ allocates all probability mass at $P_{\vec{\vartheta}}(\vec{d}^{0:t} | \vec{d}^{0:t}) = 1.0$, i.e., all other possible to-be-learned instantiations have probability zero of having been seen. Informally this means, given complete data for learning, learning parameters reduces to pure counts of observations.

As expected, learning from complete data achieves highly accurate results in representing original local CPDs (Fig. 2, 4: —•—), and one obtains nearly identical inference results (Fig. 3, 5: —•— Hellinger distance near 0). Note that, by adhering a correct prior $P(+a_{ij}^t) = 0.5$ in a dataset, not all possible activator settings are formed. This means, not all individual subset-structures are learned sequentially, but that a complete dense ADBN containing all possible substructures is learned as a bulk from data, without the need to analyze effective structures. As learning from complete data does not require a calculation of $P_{\vec{\vartheta}}(\vec{X}^{0:t^\intercal}, \vec{\mathcal{A}}^{1:t^\intercal} | \vec{d}^{0:t})$, it is linear in the size of the dataset and the number of parameters (further discussed in Sec. 5.)

4.2 Can diagonal ADBNs learn indirect influences?

In summary, no. The necessity of ADBNs is motivated by a causal design approach and the need to anticipate indirect influences. Motzek and Möller (2015b) and Example 2 demonstrate that if a diagonal (A)DBN is parametrized from a causal design perspective, spurious results are returned and indirect influences are not anticipated. However, this raises interesting question: *Can diagonal (A)DBNs learn from indirect influences and learn to anticipate, or at least, mimic them?* In order to answer this question, we repeat the previous experiment with complete datasets for four employees and

different structures: **(a)** a cyclic, dense intra-timeslice ADBN (as previously), **(b)** a dense, diagonal inter-timeslice ADBN, **(c)** a dense, diagonal inter-timeslice DBN, and **(d)** a tree-structured DBN. The difference between (b) and (c) is that (b) enforces activator constraints as done in Eq. 3.3, but (c) does not and is learned by a classic EM algorithm. To assure that sufficient datapoints are available for learning, we increase the dataset size to 50 000 datapoints.

Case (d), a tree-structured DBN as called by Ghahramani (1997), represents a previously undiscussed modeling approach, where as much intra-timeslice dependencies are modeled as possible and remaining dependencies are bent to a previous timeslice. It was not discussed previously, as it is an arbitrary decision on which dependencies are bent and which not. Generally, one could analyze activator distributions to obtain a most likely structure, which is correctly represented by intra-timeslice dependencies, and remaining dependencies are bent to a previous timeslice. However, in our domain, all structures are equally likely, which is why one has to arbitrarily choose some, e.g., choose all downward-pointing (compare Fig. 1, but for four employees) dependencies to be intra-timeslice. Then, from a reasoning perspective, a desired dependence between each and every random variable exists in one timeslice in a tree-structured DBN. Although these dependencies are causally incorrect, a learned model might be able to anticipate indirect influences, but learned CPDs are obtained by a reasoning view, which does not allow a local interpretation of them. Note that such a tree-structured DBN is the most general form of a DBN that contains the same number of parameters and dependencies as the discussed cyclic ADBN.

Experiments shown in Figure 2 show that learned CPDs heavily differ from the original models. This does not come as a surprise, as we have already noted that an naive adaption of CPDs in diagonal ADBNs cannot anticipate indirect influences, i.e., cannot learn the taintedness domain. However, a diagonal ADBN might be able to learn a *different* set of parameters which mimics an anticipation of indirect influences. For this, we compare inference results of classic filtering and smoothing problems from an original model (from which the dataset was generated) to a learned model from cases (a–d) in Figure 3. To enforce an anticipation of indirect causes, at every timestep a state variable is observed randomly. According to an effective topological ordering in a timestep t induced by $\vec{\delta}^t$, either the highest node (to enforce a forward inference of indirect causes) or the lowest (to enforce a full backward inference to indirect causes, i.e., abduction) is observed.

When validating learned non-cyclic models against observations requiring an anticipation of indirect influences, experiments shown in Figure 3 show that diagonal and tree-structured (A)DBNs deliver highly inaccurate results (Fig. 3: Hellinger distance of —+—, ·····, —+— is above 0.5), even when learned from a reasoning perspective. In fact, filtering and smoothing distributions obtained from learned non-cyclic ADBNs are almost as different from the original distribution as a distribution obtained from completely random CPDs is (Fig. 3: —▲—). Note that all experiments are performed using the same amount of data, and that learning a tree-structured DBN is as computationally expensive as learning a cyclic ADBN. A tree-structured ADBN performs slightly more accurately than a diagonal ADBN, as, at least, half of all dependencies are coincidentally

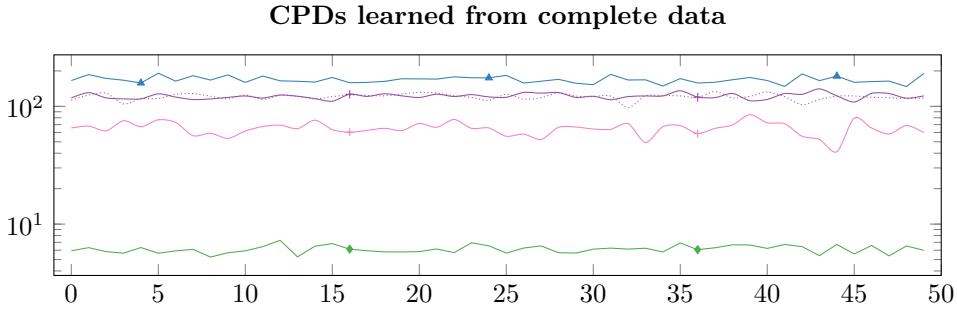

Figure 2: Absolute error (ordinate) of learned parameters for different ADBN models from complete datasets. Displayed for every ran experiment (abscissa). Classic diagonal (A)DBNs (—+—, ·····+·····) and tree-structured (A)DBNs (—+—) learn almost completely different CPDs—compare with randomly generated CPDs are given (—▲—). As expected learning a cyclic ADBN from complete datasetsin the taintedness domain (—♦—) learns nearly identical CPDs. *Semi-logarithmic plot.*

modeled causally correctly, where some indirect influences implied by observations are then handled correctly. However, a tree-structured model remains inconsequent, as it is ambiguous which dependencies are modeled in which way.

Interestingly, cases (b) and (c), i.e., a diagonal DBN with and without activator constraints, show similar results. Analyzes of learned CPDs in both cases revealed that models are learned where dependencies on other random variables are eliminated, and a process solely based on a state-variable's history is learned, i.e., all CPDs encoded $P(X_i^t|\vec{X}^{t-1\intercal}, \vec{A}_i^{t-1\,t\intercal}) = P(X_i^t|X_i^{t-1})$. This emphasizes that diagonal (A)DBNs simply do not understand indirect influences, and cannot be learned from data containing indirect influences.

The following learning cases do not consider diagonal models anymore and focus how cyclic ADBNs behave in extreme cases of incomplete data.

4.3 Hidden state variables

A common problem in classic hidden Markov models are hidden variables, where certain state variables are constantly unobservable and whose expected values must be restored during learning. In ADBNs no constant structure exists, and we extend hidden variables in ADBNs by introducing the problem of varying observability spaces, where roles of observable and unobservable variables rapidly change at every timestep. Therefore, we randomly exclude instantiations of variables $\vec{X}^t$ from each dataset at every timestep with a probability of 0.5. This means that inside one dataset $\vec{d}^{0:t}$ in every data $\vec{d}^t$ different instantiations of $\vec{x}^t$ are missing, and are sometimes missing completely. Figure 4 shows that after few EM iterations, learned local CPDs converge to a local optimum closely resembling the original CPDs (Fig. 4: —●—). Further, inference results from a learned model are very similar to results from the original model (Fig. 5: —●— below 0.1.)

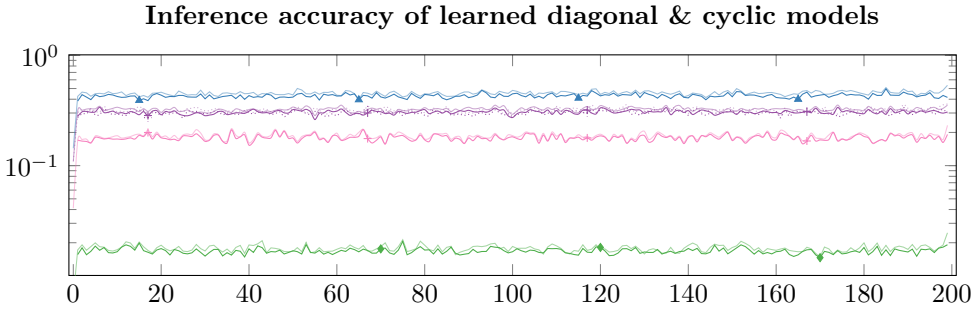

Figure 3: Inference accuracy of learned models from complete datasets. Hellinger distance (ordinate) of filtering and smoothing (lighter color) results displayed per inference-timepoint (abscissa). Classic diagonal (A)DBNs (—+—, ·····) and tree-structured (A)DBNs (—+—) achieve unsatisfying inference accuracy. A learned cyclic model (—♦—) nearly comes to the same conclusions as the original model. For reference, results of randomly generated CPDs are given (—▲—). Average of 50 experiments displayed. *Semi-logarithmic plot.*

4.4 Hidden activators

Learning datasets $\vec{d}$ with incomplete information on activators represents a novel problem of most interest. If activator instantiations are missing in a dataset, actual structural information for every timeslice is missing and must be restored based only on remaining information. Note that, at least as much information must remain to assure regularity of every learned ADBN (cf. Sec. 5). Therefore, we observe a set of activators to be deactive in our experiments, i.e., the only known structural information are *in*dependencies of a timestep and remaining structures are actively learned. Indeed, this fuses structural and parameter learning into one atomic phase. Informally, such a situation means that one cannot look at the to-be-learned data and "see" each required structure for a timeslice and a structure must be read "between the lines," i.e., is inferred live in combination with the restoration of missing instantiations of random variables.

Classic structural EM algorithms, e.g., proposed by Beal and Ghahramani (2003) or Friedman et al. (1998), score graphical models to avoid overly *complex* networks and to avoid small local optima formed by an EM approach. In order to learn ADBNs without sufficient structural information, a similar approach must be incorporated into ADBN learning. Our experiments show that without full activator instantiations, an EM approach converges towards too *simple* networks, where no dependencies are present at all, i.e., a prior of near 0 for all $P(+a_{ij}^t)$ was learned. To overcome a tendency towards too simple models, one is able to fix a prior of activators to a suitable estimation (discussed further in Sec. 5). Experimental results of Figure 4 show that with a fixed prior for activator variables, learned CPDs quickly converge to a local optimum (Fig. 4: —⋆—) and deliver inference results in the same accuracy as when dealing with hidden state variables (Fig. 5: compare —●— and —⋆—, both below 0.1).

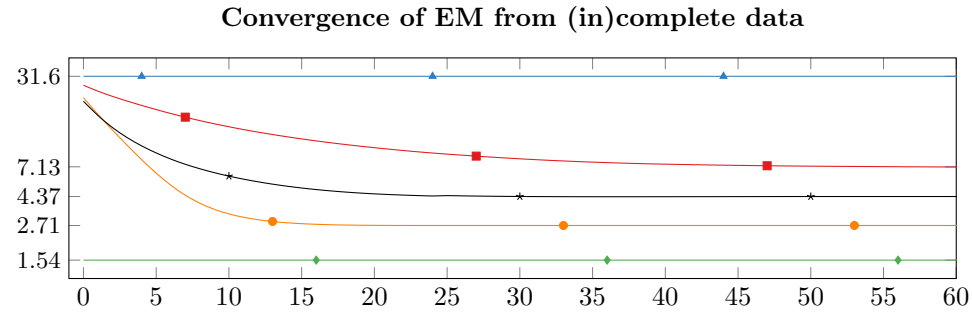

Figure 4: Convergence of learned CPDs towards the original CPDs (mean of absolute error, ordinate) over number of EM iterations (abscissa) from fully observed data (—♦—, 50 experiments), unobserved state variables X^t (—●—, 75 exp.), unobserved activator random variables A^t_{ij} (—⋆—, 70 exp.) and unobserved X^t, A^t_{ij} (—■—, 10 exp.). For reference, errors of randomly generated CPDs are given (—▲—, 50 exp.). In all cases, learned models converge towards the original models. *Semi-logarithmic plot.*

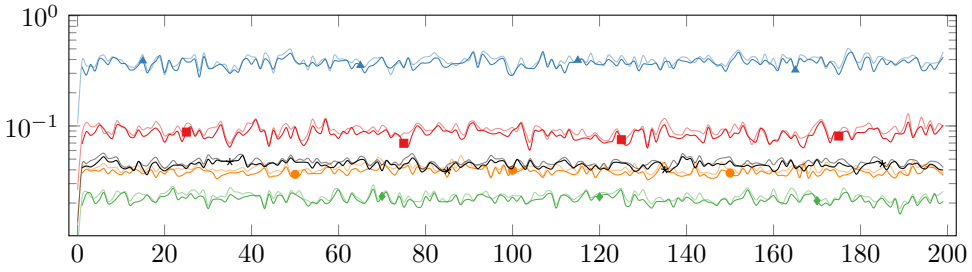

Figure 5: Inference accuracy of learned models from incomplete datasets. Hellinger distance (ordinate) of filtering and smoothing (lighter color) results display per inference-timepoint (abscissa). *Same experiments and colors as in Figure 4.* For reference, results obtained under randomly generated CPDs (—▲—) are given. Learned models nearly come to the same conclusions as the original model, even when structural information ($\vec{A}^t$) is missing. *Semi-logarithmic plot.*

4.5 Hidden activators and state variables

If datapoints contain incomplete information on activators as well as state variables, structural information is partially hidden, and effects of remaining influences between state variables are hidden as well. In this case, a structure is not known in advance and structural context information is missing as well, but information from which structural context information can be restored is (partially) missing as well. Nevertheless, multiple experiments show that the proposed EM algorithm with fixed activator priors converges to a local optima with acceptable error (Fig. 4: —■—) and which achieve

satisfactory inference results compared to the original model (Fig. 5: —■—). Note that for these experiments the length of each dataset was not increased, i.e., significantly fewer individual instantiations in each datapoint are available for learning.

5 Discussion and Related Work

Our approach for learning ADBNs is based on an EM approach for hidden variables in BNs as, e.g., presented by Ghahramani (1997) and Friedman (1997). In fact, Theorem 2 resembles a familiar "virtual" data count approach. Problems of hidden variables in addition to hidden structures have also been well studied by Friedman (1998) and have been applied to DBNs by Friedman et al. (1998), but all are subject to the assumption that a structure is knowable before random variables are instantiated. Thus, structural EM algorithms are greatly applicable to learning diagonal ADBNs, which, however, return spurious results (as demonstrated in Sec. 4 and Ex. 2), and are not applicable to learning cyclic ADBNs, as there exists no single structure on which an E-step can be performed.

A need for evolving structures over time has also been investigated by Robinson and Hartemink (2008) in the form of non-stationary DBNs (NSDBNs) and by Song et al. (2009) in the form of time-varying DBNs (TVDBNs), which both find great applications in the field of biomedicine. Robinson as well as Le Song present learning approaches for such networks from complete datasets, i.e., no hidden variables are considered, and focus on *slowly* evolving processes, where a structure remains almost constant between two timeslices. Using an ADBN has the benefit of being able to proactively model cyclic dependencies and to rapidly change a structure depending on a context at every timestep and does not require an explicit scoring and optimization of learned structures over time. A need for a sometimes rapidly switched structure has also been presented by Yoshida et al. (2005) by using Markov switching in linear models in an application towards gene networks, but veers away—unlike ADBNs—from a world-representing first-class declaration character of (D)BNs as emphasized by Pearl and Russell (2003).

Models in which a context steers an effective structure are a main point of (dynamic) Bayes multinets (DBMs) by Geiger and Heckerman (1996) and by Bilmes (2000). In DBMs an effective structure depends on one context-variable, but a general structure is bound to classic acyclicity constraints which prohibits a representation of multiple (intra-timeslice) DBNs in one model. A further consideration of roles and implications of context-specific independencies in probabilistic graphical models (PGMs) is done by Milch et al. (2005) focusing on an increased expressiveness of PGMs by the framework of (infinite) contingent Bayesian networks (CBNs). In CBNs edges are labeled with instantiations of some random variables, if the edge, i.e., dependency, is subject to a context-specific change. This edge-labeling follows a similar motivation as activator random variables do, and, similarly, an ADBN-activator-random-variable-instantiation $A_{XY} = {}^{+}a_{xy}$ can be seen as such an edge label in CBNs. Most notably, Milch et al. identify that certain domains cannot be modeled as one acyclic PGM and cyclic dependencies are required as we have as well motivated by the taintedness domain. However, Milch et al. (2005), similarly to Geiger and Heckerman (1996), introduce a novel calculus

for their context-specific PGMs, which stands in a significant contrast to ADBNs. In ADBNs no novel calculus is required, no external "outerloop" is required, and all random variables, CPDs, and the JPD as the product of all locally defined CPDs are business as usual. Moreover, CBNs make acyclicity constraints explicit by exclusive edge-labels, i.e., it is impossible to instantiate a (unrolled) cyclic CBN, whereas, in the ADBN formalism, a cyclic ADBN may be instantiated/unrolled, as regularity constraints are never enforced or required to become explicit. As Motzek and Möller (2015a) show, this circumstance is highly beneficial, as not only acyclicity is a regularity constraint. Motzek and Möller (2015b) show that show that classical algorithms such as the forward-backward algorithm remain applicable for solving filtering or smoothing problems in ADBNs. While CBNs may be able to represent ADBNs to some extent, it remains unclear whether the novelly introduced CBN-calculus still allows for these algorithms to persist and remain applicable.

The need to anticipate indirect influences arises from naturally timesliced data, evolving over time, known only in coarse grained timeslices containing hidden variables. These are contrary domains suited for Nodelman et al.'s (2002) continuous time Bayesian networks (CTBNs) and related learning approaches (Nodelman et al., 2003). Still, ADBNs and the presented learning approach can be seen as an addition to CTBNs, where observations affect variables in an uncertain coarse-grained temporal interval, while anticipating all potential implications of hidden variables during these intervals.

A cyclic ADBN contains all possible sequences formable by activator and state variables, and not solely the most likely sequences. Therefore, when learning cyclic ADBNs from data containing frequent, repeated sequences, such sequences must manifest themselves in CPDs and Markov-n ADBNs are an direct application to mining frequent sequences from incomplete, temporal datastreams. As shown by Saleh and Masseglia (2008), such frequent sequences might only exist in certain context, which is seen as a hidden variable in learning (cylic) ADBNs.

The computational complexity of the proposed learning approach for incomplete data is exponential in the number of random variables and in the number of missing instantiations in data. Please note that the same complexity, and the same amount of learning data, is required for learning a classic DBN, e.g., a tree-structured DBN and, still, DBNs show to be useful in practice. However, as we have shown, classic DBNs such as tree-structured DBNs are unable to anticipate indirect influences in extreme cases and return spurious results. Our learning approach shows a similar schema to classic learning approaches and is based on a smoothing distribution for which approximate solutions can be found via sampling sampling based approaches. If "virtual" data points are created by a sampling procedure, learning (A)DBNs reduces the complexity of counting samples. For example, learning an ADBN with 25 random variables from complete datasets with 100 000 datapoints only took 1.7s on average. Note that, for the scope of this article we used a dense intra-timeslice ADBN, i.e., the most general intra-timeslice ADBN where all activator random variables are present. The dense intra-timeslice is chosen s.t. the proposed learning approach is universally applicable, as it encodes all possible intra-timeslice DBN structures. Notwithstanding, in practice, not always all dependencies are subject to a context-specific change which will lead to fewer random variables naturally.

In this work, we consider situations where at least sufficient structural *in*dependence information remains in data. By adequate specification of local activator CPDs, it is possible to constrain ADBNs to assign zero-probabilities to non-regular instantiations by assigning a belief over possible activator constellations, i.e., structures. This means that by an adequate modeling approach, learning from data without any structural information, i.e., completely missing activator instantiations, is possible. This is closely related to the fact that a prior distribution over activator random variables must be fixed in our approach and is directly related to specify adequate heuristics finding and optimizing structures in structural EM algorithms. In fact, punishing overly complex structures, already represents a prior belief about potential structures. Note that, a distribution over potential structure candidates is directly embedded into the ADBN formalism and represents a plain additional random variable, i.e., does not require an external framework for externally creating DBN candidates. In one formalism therefore, structural- and parameter learning is collapsed to one atomic problem and integrates reasoning under multiple structure candidates. The latter idea is a novel view on work by Friedman and Koller (2003), but without external frameworks and solved in one world-representing first-class declaration of a Bayesian network.

6 Conclusion

In this paper we have derived a learning approach for ADBNs, where a structure is only knowable in a specific context of each timeslice, i.e., an instantiation of random variables. We demonstrate on a running example motivated by a cyber security application that even if parts of structure relevant information are missing and, thus, no structure is knowable, ADBNs are still able to learn model parameters which reciprocally decide an effective structure. We have shown that not even the most general form of DBNs is able to learn the taintedness-domain under the same number of dependencies, and cyclic ADBNs are causally required. As ADBNs represent superclasses of Markov-1 DBNs, parameter and structural learning of ADBNs are fused into one atomic phase of constraint parameter learning, while being able to handle hidden variables and hidden structural information.

Future work is dedicated to investigate on the applicability and merits of ADBN learning towards data mining and classic (D)BN learning under hidden variables and unknown structures.

References

Beal, M. J. and Ghahramani, Z. (2003). "The Variational Bayesian EM Algorithm for Incomplete Data: with Application to Scoring Graphical Model Structures." *Bayesian Statistics*, 7: 453–464. 16

Bilmes, J. A. (2000). "Dynamic Bayesian Multinets." In *UAI 2000: 16th Conference on Uncertainty in Artificial Intelligence, Stanford University, Stanford, California, USA, June 30 - July 3, 2000*, 38–45. 18

Boutilier, C., Friedman, N., Goldszmidt, M., and Koller, D. (1996). "Context-Specific Independence in Bayesian Networks." In *UAI 1996: 12th Conference on Uncertainty in Artificial Intelligence, Reed College, Portland, Oregon, USA, August 1-4, 1996*, 115–123. 2

Friedman, N. (1997). "Learning Belief Networks in the Presence of Missing Values and Hidden Variables." In *ICML 1997: 14th International Conference on Machine Learning, Nashville, Tennessee, USA, July 8-12, 1997*, 125–133. 18

— (1998). "The Bayesian Structural EM Algorithm." In *UAI 1998: 14th Conference on Uncertainty in Artificial Intelligence, University of Wisconsin Business School, Madison, Wisconsin, USA, July 24-26, 1998*, 129–138. 18

Friedman, N. and Koller, D. (2003). "Being Bayesian About Network Structure. A Bayesian Approach to Structure Discovery in Bayesian Networks." *Machine Learning*, 50(1-2): 95–125. 20

Friedman, N., Murphy, K. P., and Russell, S. J. (1998). "Learning the Structure of Dynamic Probabilistic Networks." In *UAI 1998: 14th Conference on Uncertainty in Artificial Intelligence, University of Wisconsin Business School, Madison, Wisconsin, USA, July 24-26, 1998*, 139–147. 2, 16, 18

Geiger, D. and Heckerman, D. (1996). "Knowledge Representation and Inference in Similarity Networks and Bayesian Multinets." *Artificial Intelligence*, 82(1-2): 45–74. 18

Ghahramani, Z. (1997). "Learning Dynamic Bayesian Networks." In *Adaptive Processing of Sequences and Data Structures, International Summer School on Neural Networks, E.R. Caianiello, Vietri sul Mare, Salerno, Italy, September 6-13, 1997, Tutorial Lectures*, 168–197. 14, 18

Glesner, S. and Koller, D. (1995). "Constructing Flexible Dynamic Belief Networks from First-Order Probabilistic Knowledge Bases." In *ECSQARU 1995: Symbolic and Quantitative Approaches to Reasoning and Uncertainty, European Conference, Fribourg, Switzerland, July 3-5, 1995*, 217–226. 1

Harsha, P. and Varma, G. (2011). "Hellinger Distance." In *Lecture Notes on Communication Complexity, School of Technology and Computer Science, Tata Institute of Fundamental Research, Mumbai, India*. 13

Jaeger, M. (2001). "Complex Probabilistic Modeling with Recursive Relational Bayesian Networks." *Annals of Mathematics and Artificial Intelligence*, 32(1-4): 179–220. 1

Milch, B., Marthi, B., Sontag, D., Russell, S. J., Ong, D. L., and Kolobov, A. (2005). "Approximate Inference for Infinite Contingent Bayesian Networks." In *AISTATS 2005: 10th International Workshop on Artificial Intelligence and Statistics, Bridgetown, Barbados, January 6-8, 2005*. 18

Motzek, A. and Möller, R. (2015a). "Exploiting Innocuousness in Bayesian Networks." In *AI 2015: 28th Australasian Joint Conference on Artificial Intelligence, Canberra, ACT, Australia, November 30 - December 4, 2015*, 411–423. 19

— (2015b). "Indirect Causes in Dynamic Bayesian Networks Revisited." In *IJCAI 2015: 24th International Joint Conference on Artificial Intelligence, Buenos Aires, Argentina, July 25-31, 2015*, 703–709. 1, 2, 3, 4, 5, 6, 7, 8, 13, 19

— (2016). "Context- and Bias-Free Probabilistic Mission Impact Assessment." Technical report, Universität zu Lübeck, Institut für Informationssysteme. Under review. URL http://www.ifis.uni-luebeck.de/~motzek/techrep-miamim.pdf 5

Motzek, A., Möller, R., Lange, M., and Dubus, S. (2015). "Probabilistic Mission Impact Assessment based on Widespread Local Events." In *NATO IST-128 Workshop: Assessing Mission Impact of Cyberattacks, NATO IST-128 Workshop, Istanbul, Turkey, June 15-17, 2015*, 16–22. 5

Murphy, K. P. (2002). "Dynamic Bayesian Networks: Representation, Inference and Learning." Ph.D. thesis, University of California, Berkeley. 1

Nodelman, U., Shelton, C. R., and Koller, D. (2002). "Continuous Time Bayesian Networks." In *UAI 2002: 18th Conference on Uncertainty in Artificial Intelligence, University of Alberta, Edmonton, Alberta, Canada, August 1-4, 2002*, 378–387. 19

— (2003). "Learning Continuous Time Bayesian Networks." In *UAI 2003: 19th Conference on Uncertainty in Artificial Intelligence, Acapulco, Mexico, August 7-10, 2003*, 451–458. 19

Pearl, J. and Russell, S. (2003). "Bayesian Networks." In Arbib, M. A. (ed.), *Handbook of Brain Theory and Neural Networks*, 157–160. MIT Press. 1, 18

Robinson, J. W. and Hartemink, A. J. (2008). "Non-Stationary Dynamic Bayesian Networks." In *NIPS 2008: 22nd Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 8-11, 2008*, 1369–1376. 18

— (2010). "Learning Non-Stationary Dynamic Bayesian Networks." *Journal of Machine Learning Research*, 11: 3647–3680. 2, 9

Russell, S. J. and Norvig, P. (2010). *Artificial Intelligence - A Modern Approach (3. internat. ed.)*. Pearson Education. 12

Saleh, B. and Masseglia, F. (2008). "Time Aware Mining of Itemsets." In *TIME 2008: 15th International Symposium on Temporal Representation and Reasoning, Université du Québec à Montréal, Canada, 16-18 June, 2008*, 93–97. 19

Song, L., Kolar, M., and Xing, E. P. (2009). "Time-Varying Dynamic Bayesian Networks." In *NIPS 2009: 23rd Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 7-10, 2009*, 1732–1740. 18

Yoshida, R., Imoto, S., and Higuchi, T. (2005). "Estimating Time-Dependent Gene Networks from Time Series Microarray Data by Dynamic Linear Models with Markov Switching." In *CSB 2005: 4th International IEEE Computer Society Computational Systems Bioinformatics Conference, Stanford, California, USA, August 8-11, 2005*, 289–298. 18