Learning Continuous-Time Bayesian Networks in Relational Domains: A Non-Parametric Approach

Shuo Yang
School of Informatics and Computing
Indiana University
Bloomington, IN 47408

Tushar Khot
Allen Institute for AI
Seattle, WA 98103

Kristian Kersting
Department of Computer Science
TU Dortmund University
Germany

Sriraam Natarajan
School of Informatics and Computing
Indiana University
Bloomington, IN 47408

Abstract

Many real world applications in medicine, biology, communication networks, web mining, and economics, among others, involve modeling and learning structured stochastic processes that evolve over continuous time. Existing approaches, however, have focused on propositional domains only. Without extensive feature engineering, it is difficult—if not impossible—to apply them within relational domains where we may have varying number of objects and relations among them. We therefore develop the first relational representation called Relational Continuous-Time Bayesian Networks (RCTBNs) that can address this challenge. It features a nonparametric learning method that allows for efficiently learning the complex dependencies and their strengths simultaneously from sequence data. Our experimental results demonstrate that RCTBNs can learn as effectively as state-of-the-art approaches for propositional tasks while modeling relational tasks faithfully.

Introduction

Modeling structured stochastic processes that evolve over time is an important and challenging task with applications in many fields ranging from surveillance to activity recognition to reliability monitoring of communication networks to treatment planning in biomedicine. Classical AI solutions such as Dynamic Bayesian networks (Murphy 2002) discretize the time into fixed intervals and perform the modeling on these intervals. While discretization is often indeed reasonable, there are domains such as medicine in which no natural discretization is available; if we discretize time too finely, the learning problem can quickly become intractable, but if we discretize time too coarsely, we lose information.

Therefore it is not surprising that models over finite spaces but across continuous time have been proposed. The most popular ones fall under the category of Continuous-Time Markov Processes. They are described by an initial distribution over the states of the model and a matrix that specifies the rate of transition between states. Its successor, Continuous-Time Bayesian Networks (CTBNs) make the problem more tractable by factorizing the rate matrix so that conditional independencies can be exploited (Nodelman, Shelton, and Koller 2002). The models can be learned efficiently from data, see e.g. (Weiss, Natarajan, and Page 2012). Without extensive feature engineering, however, it is difficult—if not impossible—to apply CTBNs to relational domains, in which e.g. there is a varying number of heterogeneous objects and relations among them. Many of today's datasets, however, are inherently relational and have no natural timeslices. Consider an Electronic Health Record (EHR). It typically contains demographic information, prescription history, lab tests, diagnoses, along with imaging data and possibly in the near future, genetic (SNPs) data as well. Another example is the human-to-X communication network where users typically call many different people, use a multitude of apps, take pictures, listen to music, and so on. Statistical Relational Learning (SRL) has been proven successful in such domains by combining the power of first-order logic and probability (Getoor and Taskar 2007). As far as we are aware, there are no continuous-time SRL approaches. It is possible to model relational CTBNs in CTPPL (Pfeffer 2009), a general purpose probabilistic programming language for processes over continuous time. However, the use of a relational language allows us to develop a more powerful structure learning approach.

Consequently, we develop *Relational Continuous-Time Bayesian Networks* (RCTBNs). They extend SRL towards modeling in continuous time by "lifting" CTBNs to relational data. The syntax and semantics are based on Bayesian Logic Programs (BLP) (Kersting and De Raedt 2002), and the use of the logical notation allows for an expressive representation that does not fix the number of features in advance yet results in a homogeneous process. Although already interesting on its own, we go one step further. Based on Friedman's functional-gradient boosting (Friedman 2001), we develop a non-parametric learning approach for RCTBNs that simultaneously learns the dependencies between the trajectories in the data and the parameters that quantify these dependencies. Our extensive experimental evaluation demonstrates that RCTBNs are comparable to state-of-the-art methods on propositional data but can handle relations faithfully, where propositional methods either fail or need to be engineered specifically for each task.

To summarize, we make the following key contributions: (1) We present the first continuous-time relational model. (2)

We develop a non-parametric learning algorithm for learning these models. (3) We prove the convergence properties of our algorithm. (4) Finally, we show that our algorithm can even model propositional data created by other methods effectively while faithfully modeling relational data.

We proceed as follows. After briefly reviewing CTBNs, we lift them to the relational case and prove that the resulting RCTBNs are homogeneous. We then show how to learn RCTBNs from data and prove convergence. Before concluding, we present our experimental results.

Background

While we will introduce relational concepts on-the-fly, let us briefly review CTBNs and functional gradient boosting.

Continuous Time Bayesian Networks (CTBNs)

As mentioned earlier, although Dynamic Bayesian Networks (DBNs) can be effective temporal models, they require discretization of time to specific slots and cannot answer queries outside these discretizations. Naturally, the question is, what rate should the DBNs use if the events all occur at distinct paces? An obvious way is to model the system at the shortest possible period (i.e. highest changing rate). However, this can lead to an increased computational cost for inference, even an exponential explosion when performing inference tasks over time. Continuous time Markov processes (CTMPs) (El-Hay et al. 2006) avoid the requirement for choosing the sampling rate by modeling the time continuously. However, since it models the transition distributions over the joint states of all variables, the size of the CTMP intensity matrix increases exponentially in the number of variables. With the forte of compact representation carried over from BNs, CTBNs (Nodelman, Shelton, and Koller 2002) model the continuous time process as the CTMP family does, but use a factored representation that compactly models joint distributions.

Following the conventional notations in BNs (Pearl 1988), we use $\mathbf{X} = \{X_1, X_2, ..., X_n\}$ to denote the collection of random variables, but now each X_i is not a discrete-valued variable but a process variable indexed by time $t \in [0, \infty)$. More precisely, $x_i(t)$ denotes the value of X_i at time t. Accordingly, the instantiation of the parents of X_i at time t is represented as $Pa_{X_i}(t)$. The instantiation of a particular set of values of $\mathbf{X}(t)$ for all t is called a *trajectory*. The key difference to BNs is the use of *conditional intensity matrices* (CIMs) instead of conditional probability distributions. Recall that BNs use parameters $P(X_i|pa_i^j)$ which indicates the conditional distributions of X_i given its parents taking the j^{th} configuration. Since CTBNs model the distribution over the variable's trajectories, a CIM induces a distribution over the dynamics of $X_i(t)$ given the values of $Pa(X_i)$ at time t (denoted by pa_i^j), which is defined as:

$$Q_{X_i|pa_i^j} = \begin{bmatrix} -q_{x_i^1|pa_i^j} & q_{x_i^1 x_i^2|pa_i^j} & \cdots & q_{x_i^1 x_i^{r_i}|pa_i^j} \\ q_{x_i^2 x_i^1|pa_i^j} & -q_{x_i^2|pa_i^j} & \cdots & q_{x_i^2 x_i^{r_i}|pa_i^j} \\ \vdots & \vdots & \ddots & \vdots \\ q_{x_i^{r_i} x_i^1|pa_i^j} & q_{x_i^{r_i} x_i^2|pa_i^j} & \cdots & -q_{x_i^{r_i}|pa_i^j} \end{bmatrix}$$

where r_i denotes the number of states the discrete variable X_i can take, $q_{x_i^k|pa_i^j} = \sum_{k' \neq k} q_{x_i^k x_i^{k'}|pa_i^j}$ and $q_{x_i^k|pa_i^j}$, $q_{x_i^k x_i^{k'}|pa_i^j} \geqslant 0$. It is factored into two components: $\mathbf{q}_{X_i}$, an *exponential distribution* over when the next transition will occur, and $\boldsymbol{\theta}_{X_i} = \mathbf{q}_{X_i^k X_i^{k'}} / \mathbf{q}_{X_i^k}(k' \neq k)$, a *multinomial distribution* over the next state. For example, $\theta_{x_i^1 x_i^2|pa_i^j}$ is the probability of X_i transitioning from state 1 to state 2 given its parents' value, pa_i^j. If X_i is in state k, it would stay in this state for an amount of time that is exponentially distributed with parameter $q_{x_i^k|pa_i^j}$, given its parents' value is pa_i^j. The expected time to transition is $\mathbb{E}_t = 1/q_{x_i^k|pa_i^j}$. Upon transitioning, X_i shifts from its k-th state to k'-th state with probability $q_{x_i^k x_i^{k'}|pa_i^j} / q_{x_i^k|pa_i^j}$. For details, we refer to first CTBN work (Nodelman, Shelton, and Koller 2002).

Instead of a single probability distribution for X_i given a specific instantiation of its parents as in BNs, CTBNs have one CIM for every configuration of its parent set. Since the number of configurations of $\mathbf{Pa}(X_i)$ equals to $v_i = \prod_{X_t \in \mathbf{Pa}(X_i)} r_t$, there are v_i conditional intensity matrices for X_i in a CTBN model, and each CIM has $r_i \times r_i$ parameters. Because of the constraint $q_{x_i^k|pa_i^j} = \sum_{k' \neq k} q_{x_i^k x_i^{k'}|pa_i^j}$, the total number of independent parameters for a CTBN model over set $\mathbf{X}$ is $\prod_{X_i \in \mathbf{X}} v_i(r_i \times r_i - r_i)$. For an individual variable X_i, its trajectory can be viewed as a conditional Markov process, which is inhomogeneous, because the intensities vary with time. It is worth noting that the CIM is not a direct function of time but rather a function of the current values of its parent variables, which also evolve with time. However, the global process of the variable set $\mathbf{X}$ is a homogeneous Markov process. In fact, any CTBN can be converted to a homogeneous Markov process by *amalgamations* (Nodelman 2007) over all its CIMs into a joint intensity matrix. Simply put, *amalgamations* are the summation over the expansions of all the CIMs over the entire variable set. Within the joint intensity matrix, all intensities corresponding to two simultaneous changes are zero because both variables cannot transit at the same instant.

Finally, indeed it is sufficient to construct a CTBN with two components: the initial distribution (denoted as $P_{\mathbf{X}}^0$) which is specified as a Bayesian network $\mathcal{B}$ over $\mathbf{X}$ and a continuous transition model, specified by: i). a directed graph $\mathcal{G}$ similar as a BN frame but possibly with cycles; ii). a set of CIMs $\mathbf{Q}_{\mathbf{X}|\mathbf{Pa}(\mathbf{X})}$, for each variable $X_i \in \mathbf{X}$.

(Relational) Functional Gradient Boosting

For learning RCTBNs, we will make use of (relational) functional gradient boosting. The standard gradient ascent approach starts with initial parameters θ_0 and iteratively adds the gradient (Δ_i) of an objective function w.r.t. θ_i. Friedman(2001) proposed an alternate approach called functional gradient boosting where the objective function is represented using a regression function ψ over the examples $\mathbf{x}$ and the gradients are derived with respect to $\psi(x)$. The key idea behind this approach is that, instead of computing the overall gradient, the gradients are approximated by computing them for each example. Each gradient term (Δ_m) is a set of a training example and a regression value given by

the gradient w.r.t $\psi_m(x_i)$, i.e., $< x_i, \Delta_m(x_i) = \frac{\partial LL(\mathbf{x})}{\partial \psi_m(x_i)} >$. To generalize from these regression examples, a regression function $\hat{\psi}_m$ (generally regression tree) is learned to fit to the gradients . The final model $\psi_m = \psi_0 + \hat{\psi}_1 + \cdots + \hat{\psi}_m$ is a sum over these regression trees.

This work has been extended to relational models (Natarajan et al. 2012; Karwath, Kersting, and Landwehr 2008; Sutton et al. 2000; Natarajan et al. 2011) by replacing the propositional trees with relational regression trees (Blockeel and Raedt 1998). If we assume the probability of an example (y_i) given its parents/neighbors($\mathbf{x_i}$) as a sigmoid of ψ, the functional gradient of each example $\langle \mathbf{x}_i, y_i \rangle$ w.r.t ψ is

$$\frac{\partial \log P(y_i; \mathbf{x_i})}{\partial \psi(y_i = 1; \mathbf{x_i})} = I(y_i = 1; \mathbf{x_i}) - P(y_i = 1; \mathbf{x_i})$$

where I is the indicator function that is 1 if $y_i = 1$ and 0 otherwise. The expression is simply the adjustment required to match the predicted probability with the true label of the example. If the example is positive and the predicted probability is smaller than 1, this gradient is positive indicating that the predicted probability should move towards 1. Conversely, if the example is negative and the predicted probability is larger than 0, the gradient is negative, driving the value the other way.

Relational CTBNs

We now introduce Relational CTBNs (RCTBNs) and then develop an efficient learning approach for them.

Syntax and Semantics

Consider the following motivating example. It is well known that the probability of developing type 2 diabetes (a hereditary disease) increases with age. Its trajectory can be modeled by CTBNs based on the expected transition time learned from data. Due to the genetic disposition, the transition probability of a person's diabetes status depends not only on his/her behavior but also on trajectories of his/her family members' diabetes status. These are in turn dynamic processes. To model relations over time faithfully, we adapt first-order logic syntax. For example, we model the family dependency using two (temporal) predicates

$$domain(Diabetes/2, discrete, [true, false])$$
$$domain(FamilyMember/2, discrete, [true, false])$$

The domain representation of $Diabetes/2$ can be interpreted as a predicate that has two arguments – the first argument running over persons and the second argument denoting the continuous time. As a binary predicate, $Diabetes$ takes $discrete$ values of $true/false$. $FamilyMember$ is a binary relation between two persons and is not temporal. Hence, it does not have time as a parameter.

To state that Mary does not have diabetes initially, we use $Diabetes(mary, t_0).\ominus$ [1]. To denote that she gets diabetes at time t_i, we use $Diabetes(mary, t_i).\oplus$. $FamilyMember(x, y)$ denotes that x is a family member of y. A sample data set could just be:

Training examples	Background Knowledge
$Diabetes(mary, t_0).\ominus$	$FamilyMember(ann, mary).$
$Diabetes(ann, t_2).\oplus$	$FamilyMember(eve, mary).$
$Diabetes(tom, t_3).\oplus$	$FamilyMember(ian, tom).$
$Diabetes(john, t_4).\oplus$	$FamilyMember(jack, bob).$
	$FamilyMember(bob, mary).$
	$FamilyMember(tom, mary).$

Now, consider the following (probabilistic) rule:

$$\forall x, t_1, \exists y, t, \quad Diabetes(x, t_1) \ominus \wedge$$
$$FamilyMember(y, x) \wedge Diabetes(y, t) \oplus \wedge t_1 < t < t_2$$
$$\Rightarrow \quad P(Diabetes(x, t_2)\oplus) = 1 - e^{-q_1(t_2 - t)}$$

It states that for all the persons (x) who do not have diabetes at time t_1, if one of his/her family members (y) develops diabetes at $t > t_1$, then the probability that x will have diabetes at time t_2 follows the exponential distribution specified at the end of the rule (with parameter q_1). Another rule is:

$$\forall x, t_1, \nexists y, t, \quad Diabetes(x, t_1) \ominus \wedge$$
$$FamilyMember(y, x) \wedge Diabetes(y, t) \oplus \wedge t_1 < t < t_2$$
$$\Rightarrow \quad P(Diabetes(x, t_2)\oplus) = 1 - e^{-q_2(t_2 - t_1)}$$

This rule contains different conditions from the first rule and says that if no family members develop diabetes in the time interval t_1 to t_2, the probability that the person will have diabetes is an exponential distribution with parameter q_2.

It is clear that, if we had used a propositional CTBN, the dimension of this feature space (the number of CIMs) would be exponential in the number of family members the individual could have. In turn, when the data set is rich in $FamilyMember$ relationships, this can greatly deteriorate the learning efficiency as observed by Weiss et.al(2012). Hence, following the standard observations inside SRL (Getoor and Taskar 2007), we exploit the ability to tie parameters using logic. More formally:

Definition A *RCTBN clause* consists of two components: a first-order horn clause that defines the effect of a set of influencing predicates on a target predicate and a conditional intensity matrix (CIM) that models the probability of transition of the target given the influences.

Definition A *RCTBN program*[2] is a set of RCTBN clauses.

Note that there could be multiple clauses with the same head, indicating the target trajectory depends on multiple parent trajectories. Second, since we are in the relational setting, there can be many instances that satisfy a single rule for a single target. For instance, in the above rule, there can be multiple family members (i.e., several instances of y) for the current person x and there can be multiple rules for predicting onset of diabetes. One is tempted to use various combining rules, see e.g. (Natarajan et al. 2009), for combining the

[1] $\ominus$ and $\oplus$ are values that we denote explicitly as against negatives in FOL. This is merely for notational simplicity.

[2] This is akin to Bayesian Logic Programs (Kersting and De Raedt 2002). Whereas, in the case of BLPs, a probabilistic clause consists of a first-order logic clause and a probability distribution associated with it, RCTBNs have a conditional intensity matrix associated with every clause, and the predicates are indexed by time.

different distributions from multiple clauses. However, for RCTBNs, the default amalgamation operation is sufficient to guarantee a homogeneous Markov process.

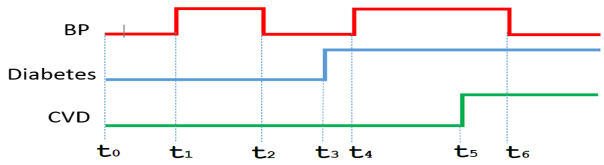
Theorem 1. *Given a set of CIMs associated with multiple RCTBN clauses, the amalgamation operation on the CIMs will result in a consistent homogeneous Markov process. The amalgamation operation is equivalent to applying the Noisy-Or combining rule to these RCTBN clauses.*

The intuition is that the amalgamation process is additive in the parameters. Similarly, we can show that the Noisy-Or operation on the exponential distribution adds the conditional intensities as well. That is, simple addition of the conditional intensities is our default combination function for the intensities (this is equivalent to applying Noisy-Or combining rule to independent transition probabilities). Nevertheless, developing equivalents of other combination functions such as weighted-mean and Noisy-And (Natarajan et al. 2009) remains an interesting direction for future research. Moreover, aggregators as used in other SRL models such as Probabilistic Relational Models (Getoor et al. 2001) can also be adopted easily in our learning formalism by allowing the search to consider aggregators of certain predicates.

Finally, it is interesting to note that unlike the directed model case of SRL models (for example BLPs), there is no necessity for explicitly checking for cycles. This is due to the fundamental assumption of the CTBN model: *two variables cannot transition out of their current state at the same instant*. So, the arcs in the transition model represent the dependencies over the trajectories, which always point from the previous time towards the current time. More precisely, $X_i \rightleftarrows X_j$ in the transition model means $X_i(t) \rightarrow X_j(t + \triangle t)$ and $X_j(t) \rightarrow X_i(t + \triangle t)$. Note that $\triangle t$ could be different in these two relations as, unlike DBNs which have a predefined sampling rate before learning the model, CTBNs model transitions over continuous time. Thus, a target variable could transit anytime based on a specific CIM during the period after the parent set changes into its current joint state and before any of them transits out of the current state. As there are no cycles in a BN within any single time slice, chain rule of BNs holds in CTBNs.

Learning RCTBNs

We now develop a non-parametric approach for learning RCTBNs from data. Note that if the training data is complete, then during the entire trajectory of $\mathbf{X}$, the values of $\mathbf{Pa}(\mathbf{X})$ are known. In turn, it is explicit at any time point which conditional intensity matrix is dominating the dynamics of $\mathbf{X}$. Thus, the likelihood of the parameters can be decomposed into the product of the likelihood of the individual parameters (Nodelman, Shelton, and Koller 2003). This parameter modularity along with the structure modularity of CTBNs allow the parameters of CTBNs to be learned locally. Besides, we notice that the likelihood function is also decomposable over individual transitions along the trajectories because of the memoryless property of the exponential distribution. These lay out the theoretical foundation for applying the non-parametric RFGB approach.



Examples	Facts
$CVD(John, t_0).\ominus$	$BP(John, t_0, low).$
$CVD(John, t_1).\ominus$	$Diabetes(John, t_0, false).$
$CVD(John, t_2).\ominus$	$BP(John, t_1, high).$
$CVD(John, t_3).\ominus$	$BP(John, t_2, low).$
$CVD(John, t_4).\ominus$	$Diabetes(John, t_3, true).$
$CVD(John, t_5).\oplus$	$BP(John, t_4, high).$
$CVD(John, t_6).\ominus$	$BP(John, t_6, low).$

Figure 1: Example trajectories of John.

Note that each training example is a segment of a trajectory that includes *not more than one* transition of the target predicate. For example, Figure 1 shows the trajectories of $BloodPressure(BP)$, $Diabetes$ and CVD for patient John. Let us assume that the goal is to predict the transition of CVD from low to high. As can be observed, every feature/predicate transits at different time points (fundamental assumption of CTBNs). This can be represented in a logical notation as we show in the bottom half of Figure 1.

Now, we first define the transition distribution from the perspective of each segment:

$$
\begin{aligned}
&\textbf{positive examples } (x_i^k \rightarrow x_i^{k'}): \\
&\quad P(x_i^k \rightarrow x_i^{k'} | pa_i^j) = \\
&\left(1 - \exp\left(-q_{x_i^k|pa_i^j} T[x_i^k x_i^{k'} | pa_i^j]\right)\right) \times \frac{q_{x_i^k x_i^{k'}|pa_i^j}}{q_{x_i^k|pa_i^j}} \\
&\textbf{negative examples } (x_i^k \rightarrow x_i^{k''}): \\
&\quad P(x_i^k \rightarrow x_i^{k''} | pa_i^j) = \\
&\left(1 - \exp\left(-q_{x_i^k|pa_i^j} T[x_i^k x_i^{k''} | pa_i^j]\right)\right) \times \frac{q_{x_i^k x_i^{k''}|pa_i^j}}{q_{x_i^k|pa_i^j}} \\
&\textbf{negative examples } (x_i^k \rightarrow x_i^k): \\
&\quad P(x_i^k \rightarrow x_i^k | pa_i^j) = \exp\left(-q_{x_i^k|pa_i^j} T[x_i^k x_i^k | pa_i^j]\right)
\end{aligned}
\tag{1}
$$

where $T[x_i^k x_i^{k'} | pa_i^j]$ is the residence time of X_i starting from being in its k-th state till transiting to its k'-th state given its parents being in their j-th joint state. We define the function value $\varphi_{x_i^k x_i^{k'}|pa_i^j} \in (-\infty, \infty)$ such that $q_{x_i^k x_i^{k'}|pa_i^j} = e^{\varphi_{x_i^k x_i^{k'}|pa_i^j}} \in [0, \infty)$. Also observe that since $q_{x_i^k x_i^{k'}|pa_i^j}$ monotonically increases with $\varphi_{x_i^k x_i^{k'}|pa_i^j}$, optimization w.r.t. $q_{x_i^k x_i^{k'}|pa_i^j}$ for CTBNs could be addressed by maximizing the loglikelihood function w.r.t. $\varphi_{x_i^k x_i^{k'}|pa_i^j}$.

To summarize, instead of maximizing the likelihood function by calculating the *sufficient statistics*—$\hat{T}[x_i^k|pa_i^j]$, the total amount of time that $X_i = x_i^k$ while $Pa(X_i) = pa_i^j$, and $M[x_i^k \rightarrow x_i^{k'}|pa_i^j]$, the total number of transitions—aggregated over the trajectories (Nodelman, Shelton, and Koller 2003), we optimize the global loglikelihood function by maximizing the individual likelihood of all the segments each of which has a weight (i.e. $\varphi_{x_i^k x_i^{k'}|pa_i^j}$) attached to it.

In the case of binary valued target predicates, the target variable has to transit to the other state upon transiting out of the current state, so $q_{x_i^k x_i^{k'}|pa_i^j}(k' \neq k) = q_{x_i^k|pa_i^j}$, hence $e^{\varphi_{x_i^k x_i^{k'}|pa_i^j}}(k' \neq k) = e^{\varphi_{x_i^k|pa_i^j}}$, and the negative examples now only contain one case where no transition happened. So, the gradient function could be derived as:

$$\frac{\partial LL}{\partial \varphi_{x_i^k x_i^{k'}|pa_i^j}} = \frac{\partial LL}{\partial q_{x_i^k x_i^{k'}|pa_i^j}} \cdot e^{\varphi_{x_i^k x_i^{k'}|pa_i^j}}$$

$$\begin{cases} \textbf{positive examples } (x_i^k \to x_i^{k'}): \\ \frac{-\exp\left(-e^{\varphi_{x_i^k|pa_i^j}} T[x_i^k x_i^{k'}|pa_i^j]\right)\left(-e^{\varphi_{x_i^k|pa_i^j}} T[x_i^k x_i^{k'}|pa_i^j]\right)}{1-\exp\left(-e^{\varphi_{x_i^k|pa_i^j}} T[x_i^k x_i^{k'}|pa_i^j]\right)} \\ \textbf{negative examples } (x_i^k \to x_i^k): \\ -T[x_i^k x_i^k|pa_i^j] \cdot \exp(\varphi_{x_i^k x_i^{k'}|pa_i^j}) \\ \quad = -e^{\varphi_{x_i^k|pa_i^j}} T[x_i^k x_i^k|pa_i^j] \end{cases} \tag{2}$$

Based on its definition in (1), the transition probability for positive examples: $prob^+ = 1 - \exp(-e^{\varphi_{x_i^k|pa_i^j}} T[x_i^k x_i^{k'}|pa_i^j])$, so the gradients function can be represented in the terms of $prob^+$ as:

$$\frac{\partial LL}{\partial \varphi_{x_i^k x_i^{k'}|pa_i^j}} = \begin{cases} \textbf{positive examples } (x_i^k \to x_i^{k'}): \\ \frac{-(1-prob^+)\ln(1-prob^+)}{prob^+} \\ \textbf{negative examples } (x_i^k \to x_i^k): \\ \ln(1-prob^+) \end{cases} \tag{3}$$

This presents the weight updates of the examples in each iteration based on the current model.

Plugging (3) into the RFGB results in a convergent learner. More precisely,

Theorem 2. *There exist global maxima for the loglikelihood function.*

Proof. The Hessian functions of the loglikelihood function w.r.t. $\varphi_{x_i^k x_i^{k'}|pa_i^j}$ equals to:

$$H = \begin{cases} \textbf{positive examples } (x_i^k \to x_i^{k'}): \\ \frac{(prob^+ + \ln(1-prob^+)) \cdot (prob^+ - 1) \cdot \ln(1-prob^+)}{(prob^+)^2} \\ \textbf{negative examples } (x_i^k \to x_i^k): \\ \ln(1-prob^+) \end{cases} \tag{4}$$

Since $prob^+ \in [0,1]$, we could derive the value ranges as: $\ln(1-prob^+) \in (-\infty, 0]$, $(prob^+ + \ln(1-prob^+)) \in (-\infty, 0]$, $prob^+ - 1 \in [-1, 0]$, and $(prob^+)^2 \in [0,1]$, hence the Hessian function w.r.t. $\varphi_{x_i^k x_i^{k'}|pa_i^j}$ would always be non-positive for any $\varphi_{x_i^k x_i^{k'}|pa_i^j}$. So, the loglikelihood is a concave function of $\varphi_{x_i^k x_i^{k'}|pa_i^j}$, which has global maxima. □

Theorem 3. *RFGB for CTBNs converges when the predictions reach the true values. In other words, RFGB for CTBNs will converge to global maxima.*

Proof. For positive examples, let $a = 1 - prob^+$, based on (3) and $L'Hôpital's$ rule, $\lim_{prob^+ \to 1} \nabla^+$ equals to

$$\lim_{a \to 0^+} \frac{-a \ln a}{1-a} = \lim_{a \to 0^+} \frac{\ln a}{1 - \frac{1}{a}} = \lim_{a \to 0^+} \frac{1/a}{1/a^2} = 0^+$$

Algorithm 1 RCTBN-RFGB: RFGB for RCTBNs

```
function RCTBNBOOST(TP, OP)
    for CT in TP do                          ▷ Iterate through target predicates
        φ_0^CT := Initial function                          ▷ Empty tree
        for 1 ≤ m ≤ M do                                    ▷ M gradient steps
            Tr := GenExamples(CT, φ_{m-1}^CT, OP)
                                                            ▷ Generate examples
            Δ_m^CT := FitRelRegressionTree(Tr, OP)
                                                            ▷ Fit trees to gradients
            φ_m^CT := φ_{m-1}^CT + Δ_m^CT                   ▷ Update model
        end for
    end for
end function
```

Algorithm 2 Example generation for RCTBNs

```
function GENEXAMPLES(CT, φ, OP)
    Tr := ∅
    for tr_i ∈ OP do                 ▷ Iterate over all transitions in OP
        Compute the residence time T of the target predicate
        Compute prob^+ = f(T, φ)     ▷ Transition probability
        if tr_i ∈ CT then
            Δ(tr_i) = (-(1-prob^+) ln(1-prob^+)) / prob^+
                                     ▷ Compute gradient of positive example
        else
            Δ(tr_i) = ln(1 - prob^+)
                                     ▷ Compute gradient of negative example
        end if
        Tr := Tr ∪ < tr_i, Δ(tr_i) >
                                     ▷ Update relational regression examples
    end for
    return Tr                                   ▷ Return regression examples
end function
```

For negative examples,

$$\lim_{prob^+ \to 0} \nabla^- = \lim_{prob^+ \to 0} \ln(1-prob^+) = 0^-$$

As shown above, $\nabla^+ \in [0, +\infty)$ and converges when the prediction equals the true value of positives (i.e. $prob^+ = 1$) while $\nabla^- \in (-\infty, 0]$ and converges when the prediction equals the true value of negatives (i.e. $prob^+ = 0$). □

The algorithm for learning RCTBNs is summarized in Alg. 1, where we use OP to denote observed predicates, TP to denote the set of target predicate transitions that we are interested in and CT, the current target transition (for example, low to high of CVD). In each gradient step, we first generate examples (denoted as Tr) based on the current φ function. We use standard off-the-shelf relational regression tree learning approaches (Blockeel and Raedt 1998) to get a regression tree that fits these gradients Tr (function $FitRelRegressionTree$). Then, we add the learned tree Δ_m^{CT} to the model and repeat this process. The examples are generated using Alg. 2. Since any non-transition can be used to generate a negative example, we can potentially have infinite negative examples (for every time point). To prevent skew and scalability issues, we generate negative examples only at time points when a certain predicate other than the target predicate has a transition. Algorithmically, we iterate over all the transitions in the trajectories. If the transition is

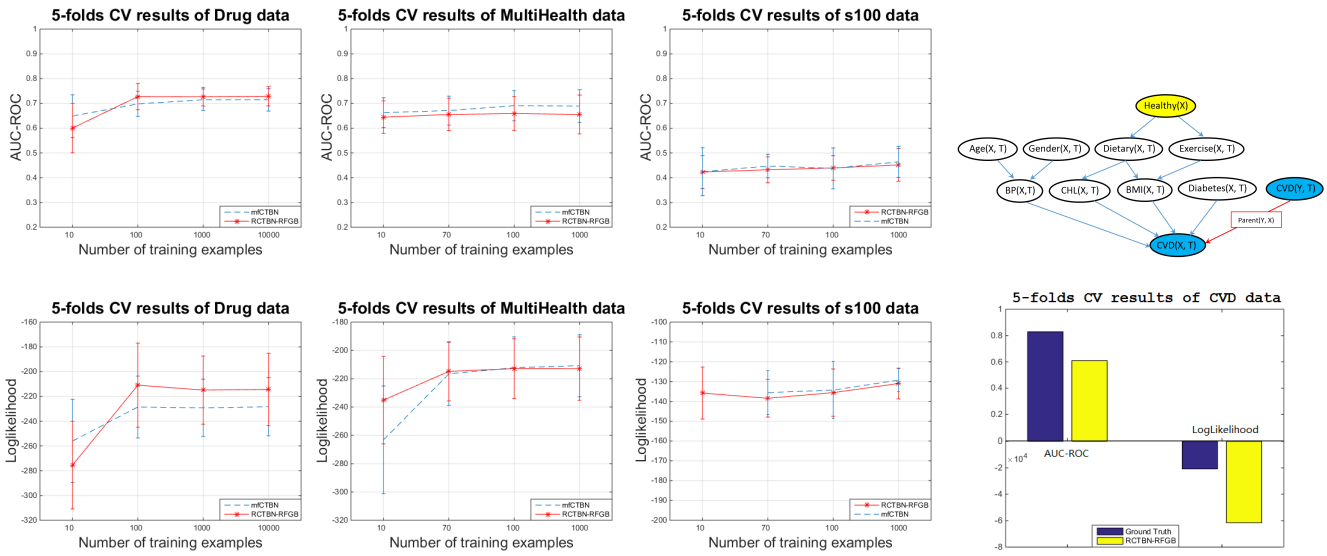


Figure 2: Left three columns, **propositional domains (Q1):** The results show no statistically significant difference between RCTBNs and state-of-the-art methods. Right column, **relational experiment (Q2): (Upper frame)** RCTBN model of CVD. **(Bottom frame)** The results show that RCTBNs can capture relations faithfully. As there is no continuous time relational model, the results are compared against "ground truth". Note that the "ground truth" does not have AUC = 1.0 because when we generated the data using forward sampling, the values are sampled based on the transition intensity calculated by $P = 1 - e^{-q*t}$. The AUC of ground truth is calculated based on these generating transition probabilities instead of the determinant values (i.e. 0 for positive examples and 1 for negative examples).

over the target predicate, we generate a regression example using the gradients for positive examples from (3). If the target predicate does not transit in this segment, we treat it as a negative example, and compute the gradient accordingly.

Experiments

Our intention is to investigate whether RCTBNs truly generalize CTBNs to relational domains by addressing the following two questions:

(Q1) How does RCTBN compare to CTBNs on propositional domains?

(Q2) Can RCTBN-RFGB learn relational models faithfully where CTBNs would fail or need extensive feature engineering ?

To this aim we evaluated RCTBN-RFGB on propositional and relational data sets and compared it to state-of-the-art where possible. More precisely, we employed three standard CTBN datasets from prior literature, i.e. the Drug model (Nodelman, Shelton, and Koller 2003), MultiHealth model (Weiss, Natarajan, and Page 2012) and S100 model (Weiss, Natarajan, and Page 2012). In the drug domain, the data is generated for length of 10, the target predicate is $JointPain(X,T)$ and other variables include the trajectories of $Eating$, $Fullstomach$, $Hungry$, $Uptake$, $Concentration$, $Barometer$, and $Drowsy$. The MultiHealth data has the target predicate $Atherosclerosis(X,T)$ and trajectories of $Gender$, $Smoking$, Age, $Glucose$, BMI, BP, $ChestPain$, $AbnormalECG$, MI, $Troponin$, $ThrombolyticTherapy$, $Arrhythmia$, $Stroke$, and HDL, for a length of 100. The S100 domain has 100 binary valued trajectories of length 2 and the target is $S100(X,T)$.

We compared RCTBN-RFGB learning approach with the state-of-the-art propositional CTBN learning approach-mfCTBN (Weiss, Natarajan, and Page 2012) on these data. Note that these are *specialized domains created by the authors of the respective papers* (Nodelman et al. for Drug and Weiss et al. for MultiHealth and S100). Our goal is to demonstrate that even for data that are not created synthetically by our model, we can still learn a comparably accurate model from them. Moreover, our formalism can handle structured inputs/outputs while these prior work cannot handle them without significant feature engineering. We ran 5-fold cross validation to generate the learning curve of the loglikelihood and AUC-ROC on the test set.

(Q1) RCTBN compares well to CTBN on propositional domains: As the left three columns in Figure 2 show, our method is comparable (no statistically significant difference at p=0.05) to the best propositional method i.e. mfCTBN on data created by mfCTBN model. This answers **(Q1)** affirmatively – RCTBN is comparable to state-of-the-art algorithms for modeling propositional dynamic processes.

(Q2) RCTBN can capture relations faithfully where CTBNs would fail or need extensive engineering: The relational data is generated by a cardiovascular disease(CVD) model shown in Figure 2(right column, top) over a duration of 50 years. We used a latent variable, $Healthy(X)$ to generate the transition patterns of dietary and exercise habit for individuals with healthy/unhealthy lifestyle. $CVD(X,T)$ is

our target predicate whose CIMs are conditioned on both the features related to the target individual and the CVD status of his/her parents. We used forward sampling (Nodelman 2007) to generate 200 trajectories for individuals whose parents are absent in the data (i.e. sample their CVD transitions based on CIMs conditioned only on white parent nodes in Figure 2(right, top)) and 200 trajectories for individuals who have one or both parents in the data (CIMs conditioned on white and blue parent nodes in Figure 2(right, top)).

For this dataset, we present the results averaged over 5 runs compared against the ground truth in Figure 2 (right column, bottom). As can be observed, our approach is comparable to the ground truth. A more compelling result is that we are able to retrieve the structure of the CVD predicate, i.e., the learned tree includes the CVD status of the parents. This answers **(Q2)** affirmatively, i.e., our proposed approach does model relations faithfully[3].

Conclusion

We proposed the first relational model that can faithfully model continuous time. Our model is inspired from the successes in the CTBNs and the SRL communities. Besides, we adapt and refine the leading learning algorithm for SRL models to the temporal case. The key innovation is that the conditional intensity is defined as a monotonic function of the RFBG function value, hence, maximizing w.r.t. function values can maximize the loglikelihood of the trajectories. Our resulting algorithm is non-parametric and can learn the dependencies and the parameters together. Our initial results are promising and show that we can indeed recover the original structure and faithfully model the relational trajectories.

Our current learning approach is discriminative and we are only learning for a single query. Extending this to generative learning and modeling jointly over multiple predicates is our next step. More applications of the algorithm to complex tasks such as Electronic Health Records, gene experimental databases, network flow modeling, and monitoring the reliability of human-to-X communication systems, among others, with asynchronous events are exciting directions. Adaptation of different types of combination functions is another important direction. Finally, improving inference is essential to scale up our learning algorithm to a large number of tasks and is also an interesting direction for future research.

Acknowledgments SY and SN gratefully acknowledge National Science foundation grant no. IIS-1343940. KK acknowledges the support of the DFG Collaborative Research Center SFB 876, project B4. The authors would like to thank David Page and Jeremy Weiss for their inputs, Jeremy Weiss for the mfCTBN code and inputs, and Christoph Ide for the discussion on communication networks.

[3]Note that our work is the first dynamic relational model to handle continuous time, see also the discussion in the introduction. Other formalisms exist to handle continuous variables but not continuous time which in our case is not a variable by itself but rather an argument of all the predicates. Hence, our relational baseline is the ground truth.

References

Blockeel, H., and Raedt, L. D. 1998. Top-Down Induction of First-Order Logical Decision Trees. *Artif. Intell.* 101(1-2):285–297.

El-Hay, T.; Friedman, N.; Koller, D.; and Kupferman, R. 2006. Continuous Time Markov Networks. In *UAI*.

Friedman, J. 2001. Greedy function approximation: A gradient boosting machine. *Annals of Statistics* 29.

Getoor, L., and Taskar, B. 2007. *Introduction to Statistical Relational Learning*. Adaptive computation and machine learning. MIT Press.

Getoor, L.; Friedman, N.; Koller, D.; and Pfeffer, A. 2001. Learning probabilistic relational models. In Dzeroski, S., and Lavrac, N., eds., *Relational Data Mining*. Springer-Verlag.

Karwath, A.; Kersting, K.; and Landwehr, N. 2008. Boosting relational sequence alignments. In *ICDM*.

Kersting, K., and De Raedt, L. 2002. Basic Principles of Learning Bayesian Logic Programs. Technical report, Institute for Computer Science, University of Freiburg.

Murphy, K. 2002. *Dynamic Bayesian Networks: Representation, Inference and Learning*. Ph.D. Dissertation, UC Berkeley, Computer Science Division.

Natarajan, S.; Tadepalli, P.; Dietterich, T. G.; and Fern, A. 2009. Learning first-order probabilistic models with combining rules. *Annals of Mathematics and AI*.

Natarajan, S.; Joshi, S.; Tadepalli, P.; Kristian, K.; and Shavlik, J. 2011. Imitation learning in relational domains: A functional-gradient boosting approach. In *IJCAI*.

Natarajan, S.; Khot, T.; Kersting, K.; Guttmann, B.; and Shavlik, J. 2012. Gradient-based boosting for statistical relational learning: The relational dependency network case. *Machine Learning*.

Nodelman, U.; Shelton, C.; and Koller, D. 2002. Continuous Time Bayesian Networks. In *Proceedings of the Eighteenth Conference on Uncertainty in Artificial Intelligence (UAI)*, 378–387.

Nodelman, U.; Shelton, C.; and Koller, D. 2003. Learning Continuous Time Bayesian Networks. In *Proc. Nineteenth Conference on Uncertainty in Artificial Intelligence (UAI)*, 451–458.

Nodelman, U. 2007. *Continuous Time Bayesian Networks*. Ph.D. Dissertation, Stanford University.

Pearl, J. 1988. *Probabilistic reasoning in intelligent systems: Networks of plausible inference*. Morgan Kaufmann Publishers Inc.

Pfeffer, A. 2009. CTPPL: A continuous time probabilistic programming language. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI)*, 1943–1950.

Sutton, R.; McAllester, D.; Singh, S.; and Mansour, Y. 2000. Policy gradient methods for reinforcement learning with function approximation. In *NIPS*.

Weiss, J. C.; Natarajan, S.; and Page, D. 2012. Multiplicative Forests for Continuous-Time Processes. In *NIPS*, 467–475.