Auxiliary Gibbs Sampling for Inference in Piecewise-Constant Conditional Intensity Models

Zhen Qin
University of California, Riverside
zqin001@cs.ucr.edu

Christian R. Shelton
University of California, Riverside
cshelton@cs.ucr.edu

Abstract

A piecewise-constant conditional intensity model (PCIM) is a non-Markovian model of temporal stochastic dependencies in continuous-time event streams. It allows efficient learning and forecasting given complete trajectories. However, no general inference algorithm has been developed for PCIMs. We propose an effective and efficient auxiliary Gibbs sampler for inference in PCIM, based on the idea of thinning for inhomogeneous Poisson processes. The sampler alternates between sampling a finite set of auxiliary virtual events with adaptive rates, and performing an efficient forward-backward pass at discrete times to generate samples. We show that our sampler can successfully perform inference tasks in both Markovian and non-Markovian models, and can be employed in Expectation-Maximization PCIM parameter estimation and structural learning with partially observed data.

1 INTRODUCTION

Modeling temporal dependencies in event streams has wide applications. For example, users' behaviors in online shopping and web searches, social network activities, and machines' responses in datacenter management can each be viewed as a stream of events over time. Models that can successfully learn the complex dependencies among events (both label and timing) allow targeted online advertising, automatic policy selection in datacenter management, user behavior modeling, or event prediction and dependency understanding in general.

[Gunawardana et al., 2011] proposed the piecewise-constant conditional intensity model (PCIM) which captures the dependencies among the types of events through a set of piecewise-constant conditional intensity functions. A PCIM is represented as a set of decision trees, which allow for efficient model selection. Forecasting via forward sampling is also simple by iteratively sampling next events based on the current history.

However, currently model selection and forecasting for PCIMs is only effective given complete data. When there are missing data, an inference method is needed to answer general queries or be employed in expectation-maximization (EM) algorithms for model selection and parameter learning. Currently, no inference algorithm has been proposed for PCIM that can condition on general evidence.

In this work, we propose the first general inference algorithm for PCIMs, based on the idea of *thinning* for inhomogeneous Poisson process [Lewis and Shedler, 1979]. This results in an auxiliary Gibbs sampler that alternates between sampling a finite set of virtual event times given the current trajectory, and then sampling a new trajectory given the set of evidences and event times (virtual and actual). Our method is convergent, does not involve approximations like fixed time-discretization, and the samples generated can answer any type of query. We propose an efficient state-vector representation to maintain only necessary information for diverging trajectories, reducing the exponentially increasing sampling complexity to linear in most cases. We show empirically our inference algorithm converges to the true distribution, permits effective query answering, and aids model selection with incomplete data for PCIM models with both Markovian and complex non-Markovian dynamics. We also show the connection between PCIMs and continuous-time Bayesian networks (CTBNs), and compare our method with an existing method on such models.

2 PREVIOUS WORK

A dynamic Bayesian network (DBN) [Dean and Kanazawa, 1988] models temporal dependencies between variables in discrete time. For systems that evolve asynchronously without a global clock, it is often not clear how timestamps should be discretized. Health records, computer server logs, and social networks are examples of asynchronous event data streams. For such systems, too slow a sampling rate would poorly represent the data, while too fast a sampling rate makes learning and inference more costly.

Continuous-time models have drawn attention recently in applications ranging from social networks [Du et al., 2013, Saito et al., 2009, Linderman and Adams, 2014] to genetics [Cohn et al., 2009] to biochemical networks [Golightly and Wilkinson, 2011]. Continuous Time Bayesian Networks (CTBN) [Nodelman et al., 2002] are homogeneous *Markovian* models of the joint trajectories of discrete finite variables, analogous to DBNs. Non-Markovian continuous models allow the rate of an event to be a function of the process's history. Poisson Networks [Rajaram et al., 2005] constrain this function to depend only on the counts of the number of events during a finite time window. Poisson cascades [Simma and Jordan, 2010] define the rate function to be the sum of a kernel applied to each historic event, and requires the modeler to choose a parametric form for temporal dependencies.

A PCIM defines the intensity function as decision trees, with internal nodes' tests mapping time and history to leaves. Each leaf is associated with a constant rate. A PCIM is able to model non-Markovian temporal dependencies, and is an order of magnitude faster to learn than Poisson networks. Applications include modeling supercomputer event logs and forecasting future interests of web search users. While PCIMs have been extended in a number of ways [Parikh et al., 2012, Weiss and Page, 2013], there is no general inference algorithms.

Inference algorithms developed for continuous systems are mainly for Markovian models or specifically designed for a particular application. For CTBNs, there are variational approaches such as expectation propagation [El-Hay et al., 2010] and mean field [Cohn et al., 2009], which do not converge to the true value as computation time increases. Sampling based approaches include importance sampling [Fan et al., 2010] and Gibbs sampling [Rao and Teh, 2011, Rao and Teh, 2013] that converge to the true value. The latter is the current state-of-the-art method designed for general Markov Jump Processes (MJPs) and its extensions (including CTBNs). It uses the idea of uniformization [Grassmann, 1977] for Markov models, similar to thinning [Lewis and Shedler, 1979] for inhomogeneous Poisson processes. We note that our inference method generalizes theirs to non-Markovian models.

3 PCIM BACKGROUND

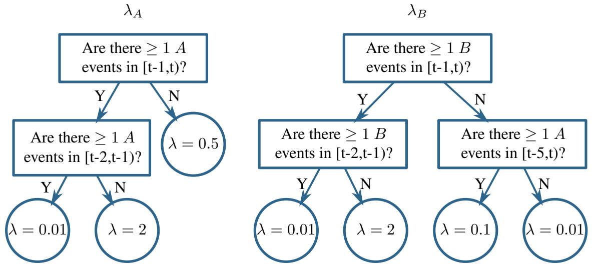
λ_A
Are there ≥ 1 A events in [t-1,t)?
Y
N
Are there ≥ 1 A events in [t-2,t-1)?
$\lambda = 0.5$
Y
N
$\lambda = 0.01$
$\lambda = 2$
λ_B
Are there ≥ 1 B events in [t-1,t)?
Y
N
Are there ≥ 1 B events in [t-2,t-1)?
Are there ≥ 1 A events in [t-5,t)?
Y
N
Y
N
$\lambda = 0.01$
$\lambda = 2$
$\lambda = 0.1$
$\lambda = 0.01$

Figure 1: Decision tree representing S and θ for events of labels A and B. Note the dependency among event labels (the rate of B depends on A). [Gunawardana et al., 2011]

Assume events are drawn from a finite label set L. An event then can be represented by a time-stamp t and a label l. An event sequence $x = \{(t_i, l_i)\}_{i=1}^{n}$, where $0 < t_1 < \ldots < t_n$. We use $h_i = \{(t_j, l_j) \mid (t_j, l_j) \in x, t_j < t_i)\}$ for the history of event i, when it is clear from context which x is meant. We define the ending time $t(y)$ of an event sequence y as the time of the last event in y, so that $t(h_i) = t_{i-1}$. A conditional intensity model (CIM) is a set of non-negative conditional intensity functions indexed by label $\{\lambda_l(t|x;\theta)\}_{l=1}^{|L|}$. The data likelihood is

$$p(x|\theta) = \prod_{l \in L} \prod_{i=1}^{n} \lambda_l(t_i|h_i;\theta)^{\mathbf{1}_l(l_i)} e^{-\Lambda_l(t_i|h_i;\theta)} \tag{1}$$

where $\Lambda_l(t|h;\theta) = \int_{t(h)}^{t} \lambda_l(\tau|h;\theta)d\tau$. The indicator function $\mathbf{1}_l(l^{'})$ is one if $l^{'} = l$ and zero otherwise. $\lambda_l(t|h;\theta)$ is the expected rate of event l at time t given history h and model parameters θ. Conditioning on the entire history causes the process to be non-Markovian. The modeling assumptions for a CIM are quite weak, as any distribution for x in which the timestamps are continuous random variables can be written in this form. Despite the weak assumptions, the per-label conditional factorization allows the modeling of label-specific dependence on past events.

A PCIM is a particular class of CIM that restricts $\lambda(h)$ to be piecewise constant (as a function of time) for any history, so the integral for Λ breaks down into a finite number of components and forward sampling becomes feasible. A PCIM represents the conditional intensity functions with decision trees. Each internal node in a tree is a binary test of the history, and each leaf contains an intensity. If the tests are piecewise-constant functions of time for any event history, the resulting function $\lambda(t|h)$ is piecewise-constant. Examples of admissible tests include

- Was the most recent event of label l?
- Is the time of the day between 6am and 9am?
- Did an event with label l happen at least n times between 5 seconds ago and 2 seconds ago?
- Were the last two events of the same label?

Note some tests are non-Markovian in that they require knowledge of more than just which event was most recent. See Fig. 1 for an example of a PCIM model.

The decision tree for label l maps the time and history to a leaf $s \in \Sigma_l$, where Σ_l is the set of leaves for l. The resulting data likelihood can be simplified:

$$p(x|S,\theta) = \prod_{l\in L}\prod_{s\in\Sigma_l} \lambda_{ls}^{c_{ls}(x)} e^{-\lambda_{ls} d_{ls}(x)}. \qquad (2)$$

S is the PCIM structure represented by the decision trees; the model parameters θ are rates at the leaves. $c_{ls}(x)$ is the number of times label l occurs in x and is mapped to leaf s. $d_{ls}(x)$ is the total duration when the event trajectory for l is mapped to s. c and d are the sufficient statistics for calculating data likelihood.

[Gunawardana et al., 2011] showed that given the structure S, by using a product of Gamma distributions as a conjugate prior for θ, the marginal likelihood of the data can be given in closed form, and thus parameter estimation can be done in closed form. Furthermore, imposing a structural prior allows a closed form Bayesian score to be used for greedy tree learning.

4 AUXILIARY GIBBS SAMPLING FOR PCIM

In this section we introduce our new inference algorithm for PCIM, called ThinnedGibbs, based on the idea of *thinning* for inhomogeneous Poisson processes. We handle incomplete data in which there are intervals of time during which events for particular label(s) are not observed.

4.1 Why Inference in PCIM is Difficult

Filling in partially observed trajectories for PCIM is hard due to the complex dependencies between unobserved events and both past and future events. See Fig. 2 for an example. While the history (the event at t) says it is likely that there should be events in the unobserved area (with an expected rate of 2), future evidence (no events in R) is contradictory: If there were indeed events in the unobserved area, those events should stimulate events happening in R.

Such a phenomenon might suggest existing algorithms such as the forward-filtering-backward-sampling (FFBS) algorithm for discrete-time Markov chains. However, there are two subtleties here: First, we are dealing with non-Markovian models. Second, we are dealing with continuous-time systems, so the number of time steps over which to propagate is infinite.

4.2 Thinning

Thinning [Lewis and Shedler, 1979] can be used to turn a continuous-time process into a discrete-time one, without using a fixed time-slice granularity. We select a rate λ^* greater than any in the inhomogeneous Poisson process and sample from a *homogeneous* process with this rate. To get a sample from the original inhomogeneous process, an event at time t is thinned (dropped) with probability $1 - \frac{\lambda(t)}{\lambda^*}$.

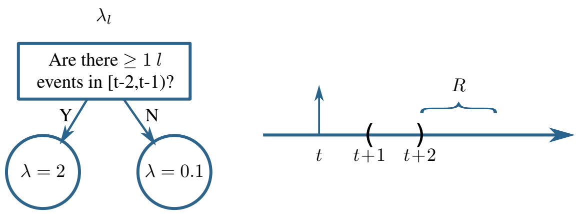

Figure 2: A simple PCIM with a partially observed trajectory. The vertical solid arrow indicates an evidence event. Areas between parentheses are unobserved. History alone indicates there should be events filled in, while the future (no events in R) provides contradictory evidence.

This process can also be reversed. If given the set of thinned event times (samples from the inhomogeneous process), extra events can be added to a sample from the original constant-rate process by sampling from a Poisson process with rate $\lambda^* - \lambda(t)$. The cycle can then repeat by thinning the new total set of times (ignoring how they were generated). At each cycle, the times (after thinning) are drawn from the original inhomogeneous process. It is this type of cycle we will employ in our sampler.

The difficulty is a PCIM is not an inhomogeneous Poisson process. Its intensity depends on the entire history of events, not just the current time. For thinning, this means that we cannot independently sample whether each event is to be thinned. Furthermore, we wish to sample from the posterior process, conditioned on evidence. All evidence (both past and future) affect the probability of a specific thinning configuration.

4.3 Overview of Our Method

To overcome both of these problems, we extend thinning to an auxiliary Gibbs sampler in the same way that [Rao and Teh, 2011, Rao and Teh, 2013] extended Markovian-model uniformization [Grassmann, 1977] (a specific example of thinning in a Markov process) to a Gibbs sampler. To do this we introduce auxiliary variables representing the events that were dropped. We call these events *virtual* events.

As a standard Gibbs sampler, our method cycles through each variable in turn. In our case, a variable corresponds to an event label. For event label l, let x_l be the sampled event sequence for this label. Let Y_l be all evidence (for l and other labels) and all (currently fixed) samples for other labels. Our goal is to sample from $p(x_l \mid Y_l)$.

Let v_l be the virtual events (the auxiliary variable) associated with l and $z_l = x_l \cup v_l$ (all event times, virtual and non-virtual). Our method first samples from $p(v_l \mid x_l, Y_l)$ and then samples from $p(x_l \mid z_l, Y_l)$. The first step adds virtual events given the non-virtual events are "correct." The second step treats all events as potential events and drops or keeps events. The dropped events are removed completely. The kept events, x_l, remain as the new sampled trajectory.

The proof of correctness follows analogously to that of [Rao and Teh, 2013] for Markovian systems. However, the details for sampling from $p(v_l \mid x_l, Y_l)$ and $p(x_l \mid z_l, Y_l)$ differ. We describe them next.

4.4 Sampling Auxiliary Virtual Events with Adaptive Rates

Sampling from $p(v_l \mid x_l, Y_l)$ amounts to adding just the virtual (dropped) events. As the full trajectory (x_l for all l) is known, the rate at any time step for a virtual event is independent of any other virtual events. Therefore, the process is an inhomogeneous Poisson process for which the rate at t is equal to $\lambda^* - \lambda_l(t|h)$ where h is fully determined by x_l and Y_l. Recall that $\lambda_l(t|h)$ is piecewise-constant in time, so sampling from such an inhomogeneous Poisson process is simple.

The auxiliary rate, λ^*, must be strictly greater than the maximum rate possible for irreducibility. We use an auxiliary rate of $\lambda^* = 2\max(\lambda(t|h))$ to sample virtual events in the unobserved intervals. This choice balances mixing time (better with higher λ^*) and computational complexity (better with lower λ^*).

A naïve way to pick λ^* is to find λ_{max}: the maximum rate in the leaves of PCIM, and use $2\lambda_{max}$. However, there could be unobserved time intervals with a possible maximum rate much smaller than λ_{max}. Using λ_{max} in those regions would generate too many virtual events, most of which will be dropped in the next step leading to computational inefficiency. We therefore use an adaptive strategy.

Our adaptive $\lambda^*(t|h)$ cannot depend on x_l (this would break the simplicity of sampling mentioned above). Therefore, we determine $\lambda^*(t|h)$ by passing (t, h) down the PCIM tree for λ_l. At each internal node, if the branch does not depend on x_l, we can directly take one branch. Otherwise, the test is related to the sampled events, and we take the maximum rate of taking both branches. This method results in $\lambda^*(t|h)$ as a piecewise-constant function of time (for the same reasons that $\lambda_l(t|h)$ is piecewise-constant).

Consider Fig. 3 as an example. When sampling event $l = A$ on the interval $[1, 5)$, we would not take the left branch at the root (no matter what events for A have been sampled), but must maximize over the other two leaves (as different x_l values would result in different leaves). This results in a $\lambda^* = 4$ over this interval, which is smaller than 6.

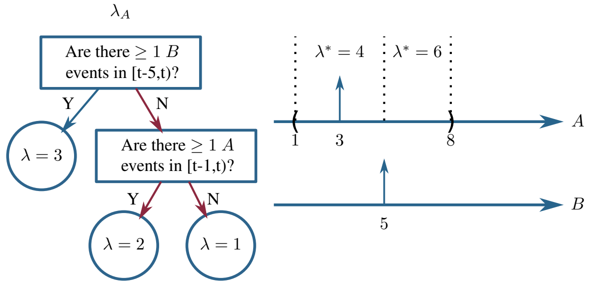

Figure 3: Adaptive auxiliary rate example. When sampling A, the branch to take at the root does not depend on unobserved events for A. If the test is related to the sampled event, we take the maximum rate from both branches. The red arrows indicate the branches to take between time $[1, 5]$, and $\lambda^* = 2 \times 2$ in that interval, instead of 6.

4.5 The Naïve FFBS Algorithm

Once these virtual events are added back in, we take z_l (the union of virtual events and "real" sampled events) as a sample from the Poisson process with rate λ^* and ignore which were originally virtual and which were originally "real." We then thin this set to get a sample from the conditional marginal over l.

The restriction to consider events only at times in z_l transforms the continuous-time problem into a discrete one. Given z_l with m possible event times $(z_{l,1}, z_{l,2}, \ldots, z_{l,m})$, let $b = \{b_i\}_{i=1}^m$ be a set of binary variables, one per event, where $b_i = 1$ if event i is included in x_l (otherwise $b_i = 0$ and the event is not included in x_l). Thus sampling b is equivalent to sampling x_l (z_l is known) as it specifies which events in z_l are in x_l. Let $Y_l^{i:j}$ be the portion of Y between times $z_{l,i}$ and $z_{l,j}$, and $b^{i:j} = \{b_k | i \leq k \leq j\}$ We wish to sample b (and thereby x_l) from $p(b \mid Y) \propto$

$$\left(\prod_i p(Y_l^{i-1:i}, b_i \mid b^{1:i-1}, Y_l^{1:i-1})\right) p(Y_l^{m:\infty} \mid b) \qquad (3)$$

where the final $Y_l^{m:\infty}$ signifies all of the evidence after the last virtual event time $z_{l,m}$ and can be handled similarly to the other terms.

The most straight-forward method for such sampling considers each possible assignment to b (of which there are 2^m). For each interval, we multiply terms from Eq. 3 of the form $p(Y_l^{i-1:i}, b_i \mid b^{1:i-1}, Y_l^{1:i-1}) =$

$$p(Y_l^{i-1:i} \mid b^{1:i-1}, Y_l^{1:i-1}) p(b_i \mid b^{1:i-1}, Y_l^{1:i}) \qquad (4)$$

where the first term is the likelihood of the trajectory interval from $z_{l,i-1}$ to $z_{l,i}$ and the second term is the probability of the event being thinned, given the past history. The first can be computed by tallying the sufficient statistics (counts and durations) and applying Eq. 2. Note that these sufficient statistics take into account $b^{1:i-1}$ which specifies events for l during the unobserved region(s), and the likelihood must also be calculated for labels $l' \neq l$ for which $\lambda_{l'}(t|h)$ depends on events from l. The second term is equal to $\frac{\lambda_l(t|h)}{\lambda^*(t)}$ if $b_i = 1$ (and $1 - \frac{\lambda_l(t|h)}{\lambda^*(t)}$ if $b_i = 0$). The numerator's dependence on the full history similarly dictates a dependence on $b^{1:i-1}$.

This might be formulated as a naïve FFBS algorithm: To generate one sample, we propagate possible trajectories forward in time, multiplying in Eq. 4 at each inter-event interval to account for the evidence. Every time we see a virtual event, each possible trajectory diverges into two (depending on whether the virtual event is to be thinned or not). By the end, we have all 2^m possible trajectories, each with its probability (Eq. 3). We sample one trajectory as the output, in proportion of the calculated likelihoods. As we explicitly keep all possible trajectories, the sampled trajectory immediately tells us which virtual events are kept, so no actual backward pass is needed.

4.6 An Efficient State-Vector Representation

The naïve FFBS algorithm is clearly not practical, as the number of possible trajectories grows exponentially with the number of auxiliary virtual events (m). We propose a more efficient state-vector representation to only keep the necessary information for each possible trajectory. The idea takes advantage of the structure of the PCIM and leads to state merges, similar to what happens in FFBS for hidden Markov models (HMMs).

The terms in Eq. 4 depend on $b^{1:i-1}$ only through the tests in the internal nodes of the PCIM trees. Therefore, we do not have to keep track of all of $b^{1:i-1}$ to calculate these likelihoods, but only the current state of such tests that depend on events with label l. For example, a test that asks "Is the last event of label l?" only needs to maintain a bit as the indicator. The test "Are there more than 3 events of label q in the last 5 seconds?" for $q \neq l$ has no state, as $b^{1:i-1}$ does not affect its choice. By contrast, a test such as "Is the last event of label q?" does depend on b, even if $q \neq l$.

As we propagate forward, we merge $b^{1:i}$ sequences that result in the same set of states for all internal tests inside the PCIM. See Fig. 4 as a simple example. Though there are 8 possible trajectories, they merge to only 2 states that we can sample from. Similar to FFBS for HMM, we need to maintain the transition probabilities in the forward pass and use them in a backward sampling pass to recover the full trajectory, but such information is also linear.

Note that this conversion to a Markov system for sampling is *not* possible in the original continuous-time system. Thinning has allows it by randomly selecting a few discrete time points, thereby restricting the possible state space to be finite.

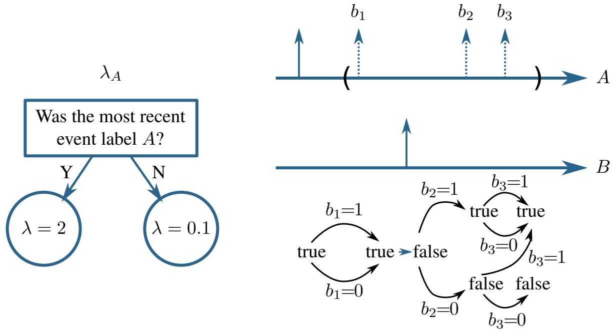

Figure 4: Dotted events are the virtual events that we sample as binary variables (b_i is 1 if event i is kept). The state diagram below the trajectory indicates the state of the test as we diverge (keep or drop a virtual event). Though there are 2^3 possible configurations, state merges can reduce the exponentially increasing complexity to linear in this case.

The state space depends on the actual tests in the PCIM model. See Tbl. 1 for the tests we currently support and their state representations. The LastStateTest and StateTest are used to support discrete finite variable systems such as CTBN, as we will use in Sec. 5 and in experiments. Note the EventCountTest was the only supported test in the original PCIM paper. We can see that for tests that only depend on the *current* time (i.e. TimeTest), the diverging history does not affect them, so no state is needed. For Markovian tests (LastEventTest and LastStateTest), we only need a Boolean variable. For the non-Markovian test (EventCountTest), the number of possible states does grow exponentially with the number of virtual events maintained in the queue. This is the best we can do and still be exact. It is much better than growing with the number of all virtual events. However, note that commonly $lag2 = 0$ and n is a small number. In this case, the state space size at any point is bounded as $\binom{m'}{n}$, where m' is the maximum number of sampled events in any time interval of duration $lag1$ (which is upper bounded by m). If n is 1, this is linear in the number of samples generated in during $lag1$ time units.

As noted above, if the test is not related to the sampled event (for example, in sampling event $l = A$ with test "are there $\geq$3 B events in the last 5 seconds"), the state of the test is null. This is because the evidence and sampled values for B (not the current variable for Gibbs sampling) can answer this test without reference to samples of l.

See Alg. 1 for the algorithm description for resampling event l. The complete algorithm iterates this procedure for each event label to get a new sample. The helper function UpdateState(s,b,t) returns the new state given the old state (s), the new time (t), and whether an event occurs at t (b). SampProbMap(M) takes a mapping from objects to positive values (M) and randomly returns one of the objects with probability proportional to the associate value. AddtoProbMap(M,o,p) checks to see if o is in M. If so, it adds p to the associated probability. Otherwise, it adds the

Table 1: Tests and their corresponding state representations.

Test	Example	State Representation	Property
TimeTest	Is the time between 6am and 9am?	Null	independent of b
LastEventTest	Is the last event A?	Boolean	Markovian
EventCountTest	Are there $>= n$ A event in $[t-lag1, t-lag2]$?	A queue maintaining all the times of A between $[t-lag2, t]$, and the most recent n events between $[t-lag1, t-lag2]$.	Non-Markovian
LastStateTest	Is the last sublabel of var $A = 0$?	Boolean	Markovian
StateTest	Is the current sublabel of var $A = 0$?	Null	independent of b

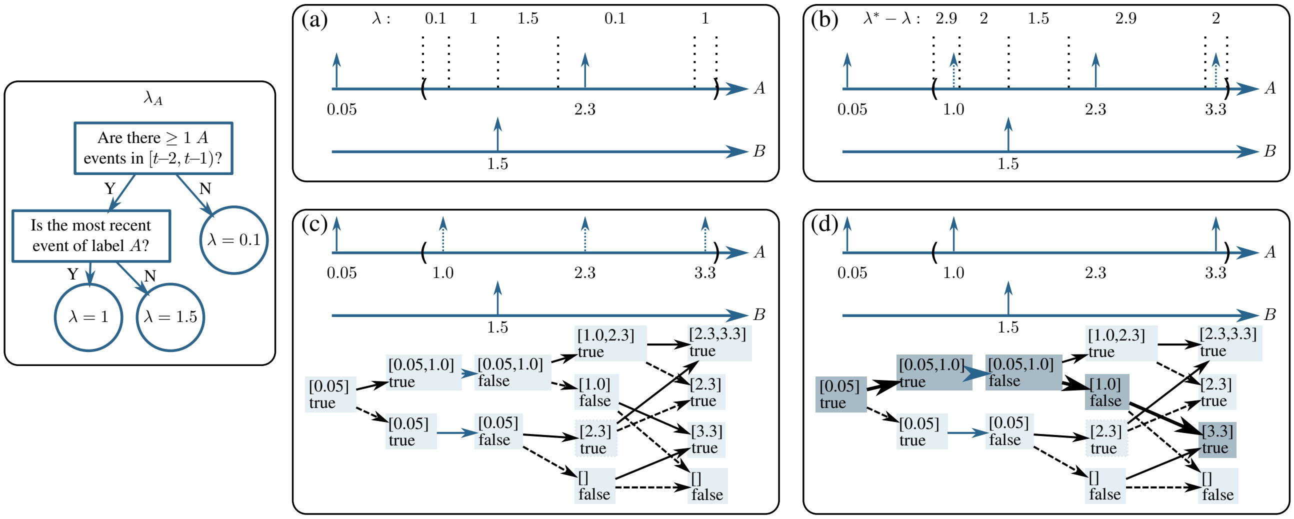

Figure 5: Extended Example, see Section 4.7

mapping $o \to p$ to M.

4.7 Extended Example

Fig. 5 shows an example of resampling the events for label A on the unobserved interval $[0.8, 3.5)$. On the far left is the PCIM rate tree for event A. Box (a) shows the sample from previous iteration (single event at 2.3). Dashed lines and λ show the piecewise-constant intensity function given the sample. Box (b) shows the sampling of virtual events. For this case $\lambda^* = 3$ for all time. $\lambda^* - \lambda$ is the rate for virtual events. The algorithm samples from this process, resulting in two virtual events (dashed). In box (c) all events become potential events. The state of the root test is a queue of recent events. The state of the other test is Boolean (whether A is more recent). On the bottom is the lattice of joint states over time. Solid arrows indicate $b_i = 1$ (the event is kept). Dash arrows indicate $b_i = 0$ (the event is dropped). Each arrow's weight is as per Eq. 4. The probability of a node is the sum over all paths to the node of the product of the weights on the path (calculated by dynamic programming). In box (d) a single path is sampled with backward sampling, shown in bold. This path corresponds to keeping the first and last virtual events and dropping the middle one.

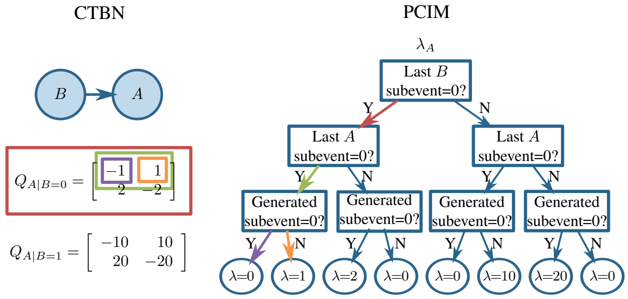

Figure 6: Explicit conversion from a CTBN to a PCIM by using specific tests. Only rates for variable A shown. The colored arrows and boxes show one-to-one correspondence of a path in the tree and an entry in the rate matrix of CTBN. Diagonal elements in the CTBN are redundant and do not need to be represented in the PCIM.

5 REPRESENTING CTBNS AS PCIMS

A non-Markovian PCIM is more general than the Markovian CTBN model. We can, therefore represent a CTBN using a PCIM. In this way, we can extend PCIMs and we can compare our PCIM method with existing methods for CTBNs.

Algorithm 1 Resampling event l

```
input: The previous trajectory $(x_l, Y_l)$
output: The newly sampled $x_l^{'}$
for each unobserved interval for $l$ do
    Find piecewise constant $\lambda^*(t|h)$ using $Y_l$
    Find piecewise constant $\lambda(t|h)$ using $x_l$, $Y_l$
    Sample virtual events $v_l$ with rate $\lambda^*(t|h) - \lambda(t|h)$
Let $z_l = x_l \cup v_l$, $m = |z_l|$, and $s_0$ be the initial state
AddtoProbMap($S_0$,$s_0$,1.0)
for $i \leftarrow 1$ to $m$ do
    for each $\{(s_{i-1}, \cdot) \to p\}$ in $S_{i-1}$ do
        $p_{keep} = p(E_{i-1:i}, b_i = 1 \mid s_{i-1}, E_{1:i-1})$
        $p_{drop} = p(E_{i-1:i}, b_i = 0 \mid s_{i-1}, E_{1:i-1})$
        $s_i^{keep} \leftarrow$ UpdateState($s_{i-1}$, true, $z_{l,i}$)
        $s_i^{drop} \leftarrow$ UpdateState($s_{i-1}$, false, $z_{l,i}$)
        AddtoProbMap($S_i$,($s_i^{keep}$, $z_{l,i}$), $p \times p_{keep}$)
        AddtoProbMap($S_i$,($s_i^{drop}$, $\emptyset$), $p \times p_{drop}$)
        AddtoProbMap($T_i(s_i^{keep})$, $(s_{i-1}, z_{l,i})$, $p \times p_{keep}$)
        AddtoProbMap($T_i(s_i^{drop})$, $(s_{i-1}, \emptyset)$, $p \times p_{drop}$)
Update $S_m$ by propagating until ending time
$x_l^{'} \leftarrow \emptyset$ and $(s_m^{'}, t) \leftarrow$ SampProbMap($S_m$)
if $t \neq \emptyset$ then $x_l^{'} \leftarrow x_l^{'} \cup \{t\}$
for $i \leftarrow m - 1$ to 1 do
    $(s_i^{'}, t) \leftarrow$ SampProbMap($T_{i+1}(s_{i+1}^{'})$)
    if $t \neq \emptyset$ then $x_l^{'} \leftarrow x_l^{'} \cup \{t\}$
return $x_l^{'}$
```

We associate a PCIM label with each CTBN variable. We also augment the notion of a PCIM label to include a sublabel. For each CTBN variable, its PCIM label has one sublabel for each state of the CTBN variable. Therefore, a PCIM event with label X and sublabel x corresponds to a transition of the CTBN variable X from its previous value to the value x. The PCIM trees' tests can also check the sublabel associated with the possible event.

We augment the auxiliary Gibbs sampler to not only sample which virtual events are kept, but also which sublabel is associated with each. This involves modifying the b_i variables from the previous section to be multi-valued. Otherwise, the algorithm proceeds the same way.

The last two tests in Tbl. 1 are explicitly for this type of sublabelled event model. We can use them to turn a conditional intensity matrix from the CTBN into a PCIM tree. Fig. 6 shows the conversion of the "twonode" model.

6 EXPERIMENTS

We implement our method as part of an open source code base, and all the code and data will be publicly available.

We perform two sets of experiments to validate our method.

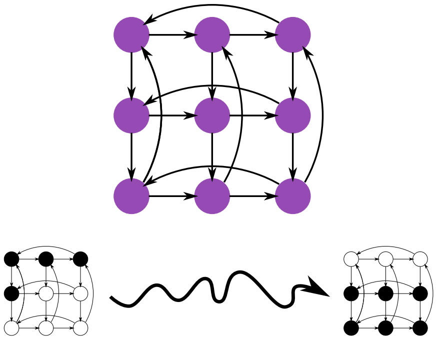

Figure 7: The toroid network and observed patterns [El-Hay et al., 2010].

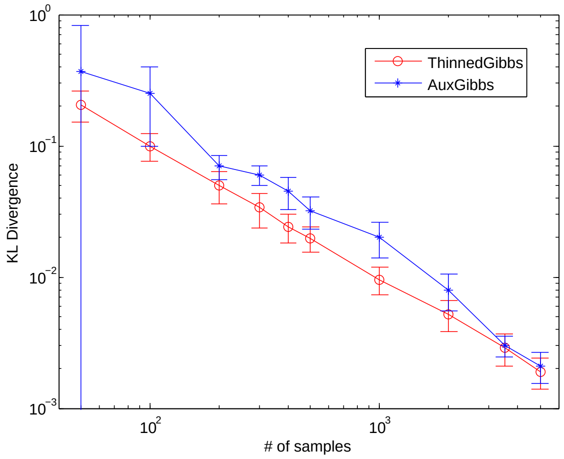

Figure 8: Number of samples versus KL divergence for the toroid network. Both axes are on a log scale.

First we perform inference with our method on both Markovian and non-Markovian models, and compare the result with the ground-truth statistics. For both we show our result converges to the correct result. Ours is the first that can successfully perform inference tasks on non-Markovian PCIMs. For the second set of experiments, we use ThinnedGibbs in EM for both parameter estimation and structural learning for a non-Markovian PCIM. Our inference algorithm can indeed help producing models that achieve higher data likelihood on holdout test data than several baseline methods.

6.1 Verification on the Ising Model

We first evaluate our method, ThinnedGibbs, on a network with Ising model dynamics. The Ising model is a well-known interaction model with applications in many fields including statistical mechanics, genetics, and neuroscience. This is a Markovian model and has been tested by several existing inference methods designed for CTBNs.

Using this model, we generate a directed toroid network structure with cycles following [El-Hay et al., 2010].

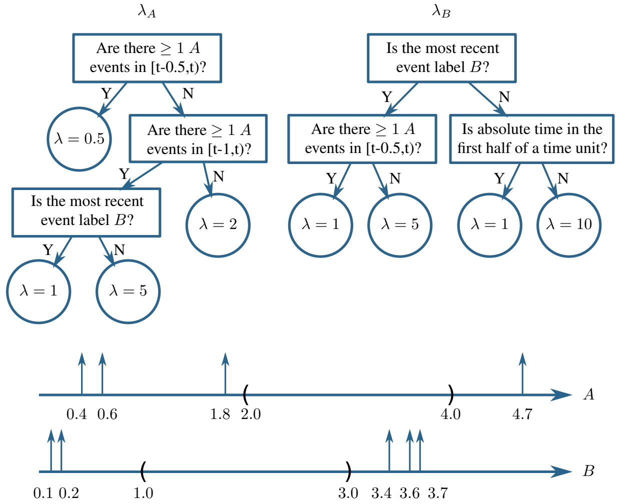
λ_A
Are there ≥ 1 A events in [t-0.5,t)?
Y
N
$\lambda = 0.5$
Are there ≥ 1 A events in [t-1,t)?
Y
N
Is the most recent event label B?
Y
N
$\lambda = 1$
$\lambda = 5$
$\lambda = 2$
λ_B
Is the most recent event label B?
Y
N
Are there ≥ 1 A events in [t-0.5,t)?
Y
N
$\lambda = 1$
$\lambda = 5$
Is absolute time in the first half of a time unit?
Y
N
$\lambda = 1$
$\lambda = 10$
A
0.4 0.6
1.8 2.0
4.0
4.7
B
0.1 0.2
1.0
3.0 3.4 3.6 3.7

Figure 9: Non-Markovian PCIM and evidence. The ending time is 5.

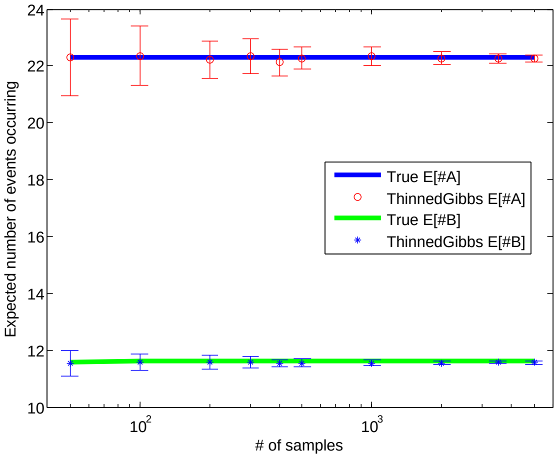

Figure 10: Number of samples versus the inferred expected number of events. The horizontal axis is on a log scale.

Nodes can take values -1 and 1, and follow their parents' states according to a coupling strength parameter (β). A rate parameter (τ) determines how fast nodes toggle between states. We test with $\beta = 0.5$ and $\tau = 2$. The network and the evidence patterns are shown in Fig. 7. The nodes are not observed between $t = 0$ and $t = 1$. We query the marginal distribution of nodes at $t = 0.5$ and measure the sum of the KL-divergences of all marginals against the ground truth. We compare with the state-of-the-art CTBN Auxiliary Gibbs method [Rao and Teh, 2013]. Other existing methods either produce similar or worse results [Celikkaya and Shelton, 2014]. We vary the sample size between 50 and 5000, and set the burn-in period to be 10% of this value. We run the experiments for 100 times, and plot the means and standard deviations.

Results in Fig. 8 verify that our inference method indeed produces results that converge to the true distribution. Our method reduces to that of [Rao and Teh, 2013] in this Markovian model. Differences between the two lines are due to slightly different initializations of the Gibbs Markov chain and not significant.

6.2 Verification on a Non-Markovian Model

We further verify our method on a much more challenging non-Markovian PCIM (Fig. 9). This model contains several non-Markovian EventCountTests. We have observations for event A at $t = 0.4, 0.6, 1.8, 4.7$ and for event B at $t = 0.1, 0.2, 3.4, 3.6, 3.7$. Event A is not observed on $[2.0, 4.0)$ and event B is not observed on $[1.0, 3.0)$.

In produce ground truth, we discretized time and converted the system to a Markovian system. Note that because the time since the last A event is part of the state, as the discretization becomes finer, the state space increases. For this small example, this approach is just barely feasible. We continued to refine the discretization until the answer stabilized. The ground-truth expected total number of A events between $[0, 5]$ is 22.3206 and the expected total number of B events is 11.6161. That is, there are about 18.32 A events and 6.62 B events in the unobserved areas. Note that if the evidence is changed to have no events these numbers drop to 1.6089 and 8.6866 respectively and if the evidence after the unobserved intervals is ignored the expectations are 22.7183 and 8.6344 respectively. Therefore the evidence (both before and after the unobserved intervals) is important to incorporate in inference.

We compare our inference method to the exact values, again varying the sample size between 50 and 5000 and setting the burn-in period to be 10% of this value. We ran the experiments 100 times and report the mean and standard deviation of the two expectations. Our sampler has very small bias and therefore the average values match the true value almost exactly. The variance decreases as expected, demonstrating the consistent nature of our method. See Fig. 10. We are not aware of existing methods that can perform inference on this type of model to which we could compare.

6.3 Parameter Estimation and Structural Learning

We further test ThinnedGibbs by using it in EM, for both parameter estimation (given the tree structure, estimate the rates in the leaves), and structural learning (learn both the structure and rates). We use Monte Carlo EM that iterates between two steps: First, given a model we generate samples conditioned on evidence with ThinnedGibbs. Second, given the samples, we treat them as complete trajectories and perform parameter estimation and structural learning, which is efficient for PCIM. We initialize the model from

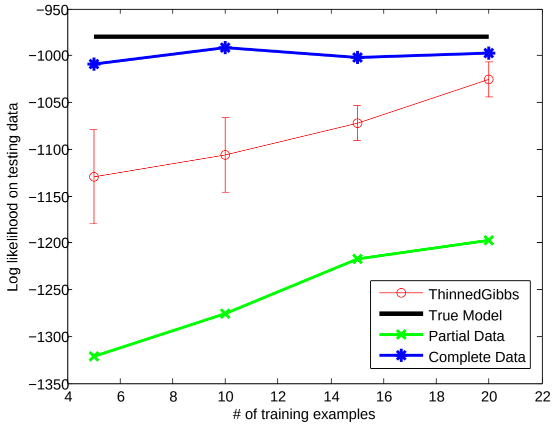

Figure 11: Parameter estimation. Testing log-likelihood as a function of the number of training samples.

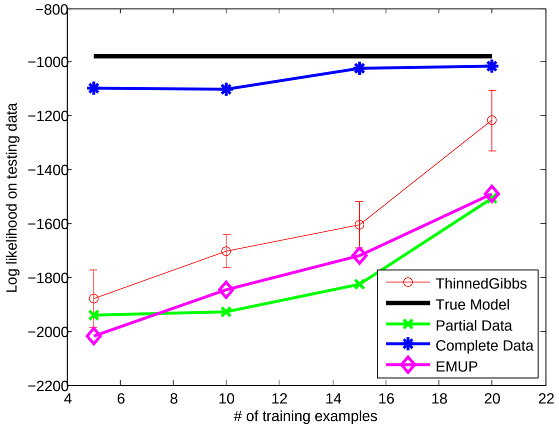

Figure 12: Structure and parameter estimation. Testing log-likelihood as a function of the number of training examples.

the partial trajectories, assuming no events occur in the unobserved intervals. EM terminates when the parameters of PCIMs in two consecutive iterations are stable (all rates change less than 10% from the previous ones), or the number of iterations surpasses 10. For structural learning, the structure needs to be the same between iterations.

We use the model in Fig. 1 and generate complete trajectories for time range $[0, 10)$. We vary the number of training samples (5, 10, 15, and 20) and use a fixed set of 100 trajectories as the testing data. For each training size, we use the same the training data for all algorithms and runs. We randomly generate an unobserved interval with length $0.6 \times T$ for both event labels. For each training sample, ThinnedGibbs fills it in to generate a new sample after burning in 10 steps. For each configuration, we run ThinnedGibbs for 5 times. We measure the data likelihood of the holdout testing data on the learned models.

For parameter estimation, we compare with the true model that generated the data, the model learned with only partial data in which we assume no events happened during unseen intervals (Partial Data), and a model learned with complete training data (Complete Data). The results are summarized in Fig. 11. We can see that the model learned by EM algorithm using ThinnedGibbs can indeed produce significantly higher testing likelihood than using only partial data. Of course, we do not do as well as if none of the data had been hidden (Complete Data).

If learning the structure, there is one other possibility: We could use the original fast PCIM learning method, but indicate (by new event labels) when an unobserved interval starts and stops. We augment the bank of possible decisions to include testing if each pseudo-events have occurred most recently. In this way, the PCIM directly models the process that obscures the data. Of course, at test time, branches modeling such unobserved times are not used. Such model should serve as a better baseline than learning from partially observed data, because it can potentially learn unobserved patterns and only use the dependencies in the observed intervals for a better model. We call this model EMUP (explicit modeling of unobserved patterns).

For structure learning, we fix the bank of possible PCIM tests as EventCountTests with $(l, n, lag1, lag2) \in \{A, B\} \times \{1, 2\} \times \{2, 3, 4, 5, 6\} \times \{0, 1, 2\}$ (omitting tests for which $lag1 \leq lag2$). For EMUP we also allow testing if currently in unobserved interval. The results are summarized in Fig. 12. We can see that EMUP does outperform models using only partial data. However, Structural EM with ThinnedGibbs still performs better. The performance gain is less than that in the parameter estimation task, probably because there are more local optimums for structural EM, especially with fewer training examples.

7 DISCUSSION AND FUTURE WORK

We proposed the first effective inference algorithm, ThinnedGibbs, for PCIM. Our auxiliary Gibbs sampling method effectively transforms a continuous-time problem into a discrete one. Our state-vector representation of diverging trajectories takes advantage of state merges and reduces complexity from exponential to linear for most cases. We build the connection between PCIM and CTBN, and show our method generalizes the state-of-art inference method for CTBN models. In experiments we validate our idea on non-Markovian PCIMs, which is the first to do so.

Our method converges to the exact conditional distribution. If the true state of the model grows exponentially, the complexity of ThinnedGibbs follows. We believe this technique could also be applied to other non-Markovian processes. The challenge lies in computing the forward-pass likelihoods when the rate function is not piecewise-constant.

Acknowledgement

This work was supported by DARPA (FA8750-12-2-0010).

References

[Celikkaya and Shelton, 2014] Celikkaya, E. B. and Shelton, C. R. (2014). Deterministic anytime inference for stochastic continuous-time Markov processes. In *ICML*. 8

[Cohn et al., 2009] Cohn, I., El-Hay, T., Kupferman, R., and Friedman, N. (2009). Mean field variational approximation for continuous-time Bayesian networks. In *UAI*. 2

[Dean and Kanazawa, 1988] Dean, T. and Kanazawa, K. (1988). Probabilistic temporal reasoning. In *AAAI*. 1

[Du et al., 2013] Du, N., Song, L., Gomez-Rodriguez, M., and Zha, H. (2013). Scalable influence estimation in continuous-time diffusion networks. In *NIPS*. 2

[El-Hay et al., 2010] El-Hay, T., Cohn, I., Friedman, N., and Kupferman, R. (2010). Continuous-time belief propagation. In *ICML*. 2, 7

[Fan et al., 2010] Fan, Y., Xu, J., and Shelton, C. R. (2010). Importance sampling for continuous time Bayesian networks. *Journal of Machine Learning Research*, 11(Aug):2115–2140. 2

[Golightly and Wilkinson, 2011] Golightly, A. and Wilkinson, D. J. (2011). Bayesian parameter inference for stochastic biochemical network models using particle Markov chain Monte Carlo. *Interface Focus*. 2

[Grassmann, 1977] Grassmann, W. K. (1977). Transient solutions in Markovian queueing systems. *Computers & Operations Research*, 4(1):47–53. 2, 3

[Gunawardana et al., 2011] Gunawardana, A., Meek, C., and Xu, P. (2011). A model for temporal dependencies in event streams. In *NIPS*. 1, 2, 3

[Lewis and Shedler, 1979] Lewis, P. A. W. and Shedler, G. S. (1979). Simulation of nonhomogeneous Poisson processes by thinning. *Naval Research Logistics Quarterly*, 26(3):403–413. 1, 2, 3

[Linderman and Adams, 2014] Linderman, S. W. and Adams, R. P. (2014). Discovering latent network structure in point process data. In *ICML*. 2

[Nodelman et al., 2002] Nodelman, U., Shelton, C. R., and Koller, D. (2002). Continuous time Bayesian networks. In *UAI*. 2

[Parikh et al., 2012] Parikh, A., Gunawardana, A., and Meek, C. (2012). Cojoint modeling of temporal dependencies in event streams. In *UAI Workshops*. 2

[Rajaram et al., 2005] Rajaram, S., Graeoel, T., and Herbrich, R. (2005). Poisson-networks: A model for structured point process. In *AIStats*. 2

[Rao and Teh, 2011] Rao, V. and Teh, Y. W. (2011). Fast MCMC sampling for Markov jump processes and continuous time Bayesian networks. In *UAI*. 2, 3

[Rao and Teh, 2013] Rao, V. and Teh, Y. W. (2013). Fast MCMC sampling for Markov jump processes and extensions. *Journal of Machine Learning Research*, 14:3207–3232. arXiv:1208.4818. 2, 3, 4, 8

[Saito et al., 2009] Saito, K., Kimura, M., Ohara, K., and Motoda, H. (2009). Learning continuous-time information diffusion model for social behavioral data analysis. In *ACML*, pages 322–337. 2

[Simma and Jordan, 2010] Simma, A. and Jordan, M. (2010). Modeling events with cascades of Poisson processes. In *UAI*. 2

[Weiss and Page, 2013] Weiss, J. and Page, D. (2013). Forest-based point processes for event prediction from electronic health records. In *ECML-PKDD*. 2