Ordinal Historical Dependence in Graphical Event Models with Tree Representations

Debarun Bhattacharjya, Tian Gao, Dharmashankar Subramanian
Research AI, IBM T. J. Watson Research Center*
{debarunb, tgao, dharmash}@us.ibm.com

Abstract

Graphical event models are representations that capture process independence between different types of events in multivariate temporal point processes. The literature consists of various parametric models and approaches to learn them from multivariate event stream data. Since these models are interpretable, they are often able to provide beneficial insights about event dynamics. In this paper, we show how to compactly model the situation where the *order* of occurrences of an event's causes in some recent historical time interval impacts its occurrence rate; this sort of historical dependence is common in several real-world applications. To overcome the practical challenge of parameter explosion due to the number of potential orders that is super-exponential in the number of parents, we introduce a novel graphical event model based on a parametric tree representation for capturing ordinal historical dependence. We present an approach to learn such a model from data, demonstrating that the proposed model fits several real-world datasets better than relevant baselines. We also showcase the potential advantages of such a model to an analyst during the process of knowledge discovery.

1 Introduction & Related Work

Modeling temporal relationships between various types (labels) of events in time-stamped streams of event occurrences is useful in a wide variety of applications, including system reliability, social networks, manufacturing processes, retail, healthcare, politics, and finance. It is well known that multivariate event streams in continuous time can be modeled as samples from a *marked (or multivariate) point process* [Aalen, Borgan, and Gjessing 2008; Cox and Lewis 1972] – a stochastic process that involves counting processes for event labels. Each label is associated with a *conditional intensity* function that determines its rate of occurrence at any time given historical occurrences of its causal event labels.

The literature on multivariate point processes is vast and varied, spanning parametric as well as neural models for specifying how conditional intensity rates vary with historical occurrences. Parametric models that are most popular are primarily of two types. One stream of research explores multivariate Hawkes processes with jumps in intensity at historical arrival epochs that decay over time [Bacry, Mastromatteo, and Muzy 2015; Etesami et al. 2016; Xu, Luo, and Zha 2017; Yu et al. 2020]. Another line of work considers models where the conditional intensity rate is a piece-wise constant function of historical occurrences [Gunawardana, Meek, and Xu 2011; Parikh, Gunawardana, and Meek 2012; Bhattacharjya, Subramanian, and Gao 2018] . Other parametric approaches have also been pursued [Simma et al. 2008], as have approaches involving neural architectures [Du et al. 2016; Mei and Eisner 2017; Xiao et al. 2017; Gao et al. 2020].

All the aforementioned temporal models fit within the high-level framework of *graphical event models (GEMs)* [Didelez 2008; Meek 2014], which are graphical representations of marked point processes that explicitly indicate which labels' historical occurrences have a direct influence on the process dynamics of any particular event label. Such a graphical framework improves interpretability and can be tremendously insightful in applications where knowledge discovery is the key task. The piece-wise constant family has an additional advantage beyond interpretability; given a graph, learning the conditional intensity parameters can typically be done efficiently using summary statistics [Gunawardana, Meek, and Xu 2011; Bhattacharjya, Subramanian, and Gao 2018].

Recent work within the piece-wise constant family of GEMs has shown the promise of *ordinal* historical dependence, where an event label's arrival rate is determined by the recent historical order in which its underlying causal events have occurred [Bhattacharjya, Gao, and Subramanian 2020]. However, the state-of-the-art model and its associated learner cannot adequately handle the super-exponential number of potential orders as a function of the causes. This greatly limits the model's applicability on practical datasets as it is often unable to identify more than two or three parents. Furthermore, the model assumes access to user-provided inputs which may be unavailable without domain expertise. In this paper, we introduce a novel tree-based representation for conditional intensity parameters that compactly captures ordinal historical dependence, as well as an associated learner that is effective and more insightful for real-world datasets.

Motivation. We motivate the model with the help of an illustrative example involving social unrest. Consider the (ordinal) graphical event model in Figure 1(a) with 4 types of recurring events: natural disasters (D) such as earthquakes or

*DISTRIBUTION STATEMENT A. Approved for public release: distribution unlimited

Copyright © 2021, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

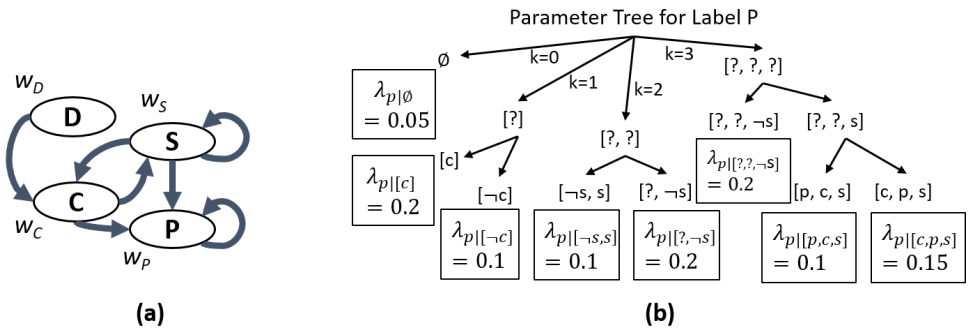

Figure 1: (a) Structure of an illustrative ordinal graphical event model (OGEM) for social unrest with 4 nodes (event labels): disaster (D), crisis (C), stimulus (S) and protest (P). Windows for each node are also shown. (b) Illustrative tree parameter representation for protest (P) events, with conditional intensities for every leaf order representation.

epidemics, socio-economic crises (C) such as food shortages, government stimulus packages (S) and protests (P). In this example, a disaster increases the rate of crisis events, while a government stimulus might be able to dampen crisis events as well as protests. Stimulus and protest events have self-loops, with dampening and amplifying effects respectively.

Figure 1(b) shows an illustrative parametric representation for the rate of protests (P) depending on the order of its causes {C,P,S} – its parents in Figure 1(a) – in some recent time window w_P (such as a week). The data structure is a tree that is split based on the number of preceding event labels, ranging from $k = 0$ where nothing happens to $k = 3$ where all 3 parent labels occur in some order. Each node in the tree represents a grouping of orders of subsets of {C,P,S} with a fixed length k, where slots are either filled or available or restricted; we refer to them as *order representations*. For instance, $[?, \neg s]$ covers orders where the first label can be any parent but the second cannot be S. The parameters at the leaves compactly capture the effect of different equivalent groupings of orders.

In this example, if only a single event ($k = 1$) occurs in the window, then whether or not it is a crisis event ($[c]$ vs $[\neg c]$) impacts the rate λ_p. For $k = 2$ or $k = 3$, whether or not a stimulus is the most recently occurred label is crucial to determine the rate. There is an additional nuance for $k = 3$ where it also matters if a stimulus is provided immediately after a crisis ($[p, c, s]$ vs $[c, p, s]$); if not, the government is perceived to be slow to respond. Note that the representation allows for complex ordinal historical dependence while enabling parameter sharing; here there are 8 parameters rather than one for each of the 16 possible orders of all subsets of {C,P,S}. The reduction in the number of parameters is crucial for learning, as it enables the detection of longer causal orders/chains as influencers of a label's arrival rate. Further details regarding the order representations, how orders are determined from data when there could be multiple occurrences of the same label, model learning, etc. will be covered in later sections.

Summary of Contributions. Our main contribution is a new GEM that deploys an ordinal tree representation for conditional intensities. The representation is inspired by conditional probability distributions in Bayesian networks [Pearl 1988] that embody context-specific independences [Boutilier et al. 1996; Geiger and Heckerman 1996; Poole and Zhang 2003]. The novelty lies in the use of a tree for sharing *ordinal conditional intensity parameters* in graphical event models, making it even more important in practice due to the super-exponential number of orders of subsets of a set of parents and the widespread availability of event stream data. Prior work that has used trees in event models considers whether events occur in user-provided historical basis functions [Gunawardana, Meek, and Xu 2011]; in contrast, here we propose an approach that groups parameters for historical orders. We describe a learning algorithm and show that the proposed model fits many real-world datasets better than relevant baselines. The main benefit however comes from additional insights that the proposed model could provide to an analyst; we briefly showcase some of these by highlighting differences with baseline models that often learn sparser graphs and do not distinguish between historical orders.

2 Background

An event stream is a sequence of time-stamped events, $\mathcal{D} = \{(l_i, t_i)\}_{i=1}^{N}$, where event label l_i occurs at time t_i. l_i belongs to label set $\mathcal{L}$ with cardinality $M = |\mathcal{L}|$, and time $t_i \in \mathbb{R}^+$ lies between $t_0 = 0$ and final time $t_{N+1} = T$. All results in the paper extend to multiple independent streams.

Any event stream in continuous time, under reasonable regularity conditions, can be modeled as a marked point process. The dynamics of event occurrences are captured by conditional intensity functions, which measure the rate at which an event label occurs. In general, the conditional intensity for event label X at time t can be written as a function of the *history* at that time, h_t, i.e. it is denoted $\lambda_x(t|h_t)$ where $h_t = \{(l_i, t_i) : t_i < t\}$ includes all preceding events at time t. Not all event labels may however necessarily influence X's rate directly, which can be captured using the notion of process independence [Didelez 2008] in a graphical representation:

Definition 1. *A **graphical event model** is a framework for representing a marked point process over event labels $\mathcal{L}$. It includes:*

- *A directed graph $\mathcal{G}$ with a node for every event label.*
- *Conditional intensity functions for each label $X \in \mathcal{L}$ at any time t that depend only on historical occurrences of its parent labels in $\mathcal{G}$. Thus $\lambda_x(t|h_t) = \lambda_x(t|[h(\mathbf{U})]_t)$, where $\mathbf{U}$ are X's parents and $[h(\mathbf{U})]_t$ is the restricted history, $[h(\mathbf{U})]_t = \{(l_i, t_i) : t_i < t, l_i \in \mathbf{U}\}$.*

We remind the reader that GEMs are merely a family of models; to specify the model entirely, it is necessary to completely describe how the conditional intensities vary with historical occurrences. Furthermore, one cannot consider arbitrary historical dependence for learnability; some sort of assumptions (such as around stationarity) are required. In the next section, we discuss historical dependencies where the recent historical order of occurrences of an event label's causes (parents in the GEM graph) impacts its arrival rate.

3 Ordinal Historical Dependence

We are interested in a GEM that captures ordinal historical dependence, but since each parental label may occur several times in history in a recurring event stream, a masking function [Bhattacharjya, Gao, and Subramanian 2020] is required, to retain only distinct label occurrences from history, thereby determining the active parental order at any time.

Definition 2. *A **masking function** $\phi(\cdot)$ converts an event tuple sequence into a sub-sequence where no label is repeated. Formally, $\phi(\cdot)$ takes as input an event sequence $s = \{(l_j, t_j)\}$ and returns $s' = \{(l_k, t_k) \in s : l_k \neq l_m \text{ for } k \neq m\}$. It induces a unique event label order on the input sequence s, through a temporal ordering of the distinct labels in s'.*

We consider two cases of masking function $\phi(\cdot)$ due to their simplicity and potential applicability: the 'first' and 'last' cases, depending on whether only the first or last occurrence of an event label in the input sequence is retained. For example, the 'first' and 'last' masking functions convert sequence $[(S, 10), (C, 15), (S, 30)]$ into label orders $[S, C]$ and $[C, S]$ respectively. These masking functions are suitable for datasets where events do not occur too often, but other potential masking functions may also be appropriate.

Definition 3. *An **order instantiation** o for a set of labels $\mathbf{Z}$ is a permutation of a subset of $\mathbf{Z}$. The order instantiation at time t in an event dataset $\mathcal{D}$ over a preceding time window w is determined by applying masking function $\phi(\cdot)$ to events restricted to labels $\mathbf{Z}$ occurring within $[\max(t - w, 0), t)$.*

In the remainder of the section, we distinguish between two kinds of ordinal graphical event models (OGEMs) based on the representation of the conditional intensity functions.

3.1 Tabular Ordinal Graphical Event Model

We borrow terminology from the Bayesian network literature [Koller and Friedman 2009]: when there is a corresponding conditional intensity parameter for every order instantiation of a node's parents, we refer to the representation as *tabular*. This version of the model in totality is defined as:

Definition 4. *A **tabular ordinal graphical event model** [Bhattacharjya, Gao, and Subramanian 2020] for a masking function $\phi(\cdot)$ and event label set $\mathcal{L}$ includes:*

- *A directed graph $\mathcal{G}$ with a node for every event label.*
- *Windows for every node in $\mathcal{G}$, $\mathcal{W} = \{w_X : X \in \mathcal{L}\}$.*
- *Conditional intensity rate parameters Λ, one for every node and order instantiation o with respect to the node's parents, $\Lambda = \{\Lambda_X : X \in \mathcal{L}\} = \{\lambda_{x|\mathbf{o}} : X \in \mathcal{L}, \forall \mathbf{o}\}$.*

Figure 1(a) shows an illustrative OGEM graph with 4 nodes, along with the windows for every node. Unlike other parametric models, an OGEM is intended to explicitly represent different rates for different orders of a node's parents. Unfortunately, a tabular OGEM is super-exponential in its parametric complexity since it has a rate parameter for each order instantiation; for a node X with parents $\mathbf{U}$, there are $\sum_{i=0}^{|\mathbf{U}|} \frac{|\mathbf{U}|!}{i!}$ possible orders. This is problematic from the perspective of learning for several reasons; for instance, it is likely that most order instantiations will never be observed in the data. Thus, for OGEMs to be practically useful, a more compact parameter representation is crucial.

3.2 Tree Ordinal Graphical Event Model

Figure 1(b) provides an example where multiple order instantiations share a common rate parameter. For instance, $[s, p, c]$, $[p, s, c]$, $[c, s, p]$ and $[s, c, p]$ are four distinct order instantiations of size $k = 3$ for node P with parents {C,P,S} that have a common rate $\lambda = 0.2$. However, it is not obvious how to formalize parameter compression relative to the exhaustive ordinal tabular representation. We propose a novel approach using what we refer to as an *order representation*.

Definition 5. *An **order representation** $\mathbf{r}$ of length $k \leq |\mathbf{U}|$ for a set of labels $\mathbf{Z}$ is a sequence of k slots (i.e. positions in the sequence) that are either filled with a label in $\mathbf{Z}$ or declared not to belong to a subset of $\mathbf{Z}$ (including $\emptyset$). It is **feasible** if there is at least one order instantiation $\mathbf{o}$ of $\mathbf{U}$ that is compatible with $\mathbf{r}$, i.e. satisfies the conditions in each slot.*

An order representation $\mathbf{r}$ groups order instantiations. While denoting slots in $\mathbf{r}$, we use the symbol '?' to denote no restrictions, and negation '$\neg$' to imply a specified restriction. For instance, in Figure 1(b), the order representation $[?, ?, \neg s]$ for orders of length $k = 3$ keeps the first two slots unspecified and puts a restriction on the third slot disallowing event label S, thereby enabling a common rate parameter for each of $[s, p, c]$, $[p, s, c]$, $[c, s, p]$ and $[s, c, p]$ since it is a leaf. A tree OGEM is one where parameters are mutually exclusive and collectively exhaustive order representations:

Definition 6. *A **tree ordinal graphical event model** for a masking function $\phi(\cdot)$ and event label set $\mathcal{L}$ includes:*

- *A directed graph $\mathcal{G}$ with a node for every event label.*
- *Windows for every node in $\mathcal{G}$, $\mathcal{W} = \{w_X : X \in \mathcal{L}\}$.*
- *Conditional intensity rate parameters Λ, one for every node and set of feasible, mutually exclusive and collectively exhaustive order representations r (with respect to any subset of the node's parents), $\Lambda = \{\Lambda_X : X \in \mathcal{L}\} = \{\lambda_{x|\mathbf{r}} : X \in \mathcal{L}, \forall \mathbf{r}\}$.*

While it is not necessary for the conditional intensity parameters in a tree OGEM to be organized as leaves in a tree, it is convenient to do so, to enforce mutually exclusive and collectively exhaustive leaf order representations, in which case we denote them as $\mathbf{r_L}$. Indeed, the use of

slot-specification and slot-restriction complement each other and both are necessary to enable splitting an internal node in an order representation tree. For example, splitting the node order representation $[?, ?, ?]$ of length $k = 3$ in Figure 1(b) produces two children which respectively have a slot-specification and slot-restriction in the third slot. We describe tree learning in further detail in the next section.

Note that a fully specified order representation of length k with no further degree of freedom is an order instantiation of length k, but the flexibility from allowing lack of full specification enables coverage of variations of prior models in the literature. The following result highlights some generality of the tree representation, showing that it encompasses tabular OGEMs as well as an important special case of another parametric GEM. All theorem proofs are in Appendix A[1].

Theorem 7. *A tabular OGEM and a proximal GEM (PGEM) [Bhattacharjya, Subramanian, and Gao 2018] where edges from parents have the same window are special cases of a tree OGEM.*

4 Learning

We present an approach for learning a tree OGEM $\{\mathcal{G}, \mathcal{W}, \Lambda\}$ from an event dataset $\mathcal{D}$. We focus on learning the tree parameter representation Λ and a heuristic for windows $\mathcal{W}$. Details around learning graph $\mathcal{G}$ and computation of summary statistics are relegated to Appendix B as they have been studied in prior work [Bhattacharjya, Gao, and Subramanian 2020]. Since all GEMs are potentially cyclic, one can learn the model for each event label separately.

4.1 Learning Tree Parameter Representations

Suppose that an event label X's parents $\mathbf{U}$ and window w_X are known. We take a score-based approach to learn a tree parameter representation, where the tree consists of a sub-tree for each order representation length k from 0 to $|\mathbf{U}|$.

Computing the log likelihood and a score that adjusts for model complexity is reasonably straightforward, at any stage of tree construction. For event label X in a tree OGEM with leaf order representations $\mathbf{r_L}$ and corresponding intensities $\lambda_{x|\mathbf{r_L}}$, its log likelihood for an event dataset $\mathcal{D}$ is:

$$\text{logL}_X(\mathcal{D}) = \sum_{\mathbf{r_L}} \left(-\lambda_{x|\mathbf{r_L}} D(\mathbf{r_L}) + N(x; \mathbf{r_L}) \log(\lambda_{x|\mathbf{r_L}})\right), \tag{1}$$

where $N(x; \mathbf{r_L})$ is the number of times X is observed in the dataset and the order instantiation is consistent with the order representation $\mathbf{r_L}$ in the preceding window w_X, and $D(\mathbf{r_L})$ is the duration over the entire time period where order instantiations consistent with the condition $\mathbf{r_L}$ hold. Counts and durations depend on the window w_X but this is hidden in the notation for simplicity. We use the Bayesian information criterion (BIC) score:

$$S_X(\mathbf{U}; \Lambda_X; \mathcal{D}) = \text{logL}_X(\mathcal{D})^* - \gamma \frac{|\Lambda_X|}{2} \log(T), \tag{2}$$

where $\text{logL}_X(\mathcal{D})^*$ is the log likelihood for X from Equation (1) computed at the maximum likelihood estimates for rates $\left(\hat{\lambda}_{x|\mathbf{r_L}} = \frac{N(x;\mathbf{r_L})}{D(\mathbf{r_L})}\right)$, $|\Lambda_X|$ is the number of conditional intensities for X in the model, γ is a penalty weight on the complexity (second) term, typically set to 1, and T is the time horizon of the event dataset.

[1]Appendices can be found in the arXiv version of the paper.

Algorithm 1 Learn Tree Representation

procedure OPTTREE(event label X, parents $\mathbf{U}$, window w_X, masking function $\phi(\cdot)$, dataset $\mathcal{D}$)
  Initialize root of tree $\mathcal{T}$
  for k from 0 to $|\mathbf{U}|$ **do**
    Learn optimal subtree $\mathcal{T}_k$ from procedure 'OptSubtree'
    Make its root a child of the tree root
  return Optimal tree $\mathcal{T}$

procedure OPTSUBTREE(event label X, parents $\mathbf{U}$, window w_X, masking function $\phi(\cdot)$, dataset $\mathcal{D}$, subtree length k)
  Initialize representation list $\mathcal{R}$, tree $\mathcal{T}_k$ and model information for representations $\mathcal{I}$ as empty
  Set root of subtree as $r = [?, ?, \cdots]$ (k times)
  Add r to list $\mathcal{R}$ and tree $\mathcal{T}_k$
  Compute all model information (summary stats, lambdas, LL and score) for the root; store in $\mathcal{I}$
  while $\mathcal{R}$ not empty **do**
    Choose any representation r in $\mathcal{R}$ and determine all feasible splits by filling a single slot
    for both children r_C in each feasible split of r **do**
      if $r_C \in \mathcal{I}$ **then**
        Retrieve model information from $\mathcal{I}$
      else
        Compute all model information; store in $\mathcal{I}$
    Consider feasible split with maximum total score
    if feasible split and score improvement from this split over parent > 0 **then**
      Make parent r an internal node of tree $\mathcal{T}_k$
      Add both r_C from this split to list $\mathcal{R}$
    else
      Remove parent r from list $\mathcal{R}$; make it a leaf node
  return Optimal sub-tree $\mathcal{T}_k$ for this k; Model info. $\mathcal{I}$

Algorithm 1 outlines a greedy approach for growing subtrees and therefore determining parameters at the leaves. Each sub-tree starts with the root order representation that groups all order instantiations of same length. A list of nodes $\mathcal{R}$ in the tree is maintained, as is a data stucture $\mathcal{I}$ for storing information such as log likelihoods and scores. For each node in $\mathcal{R}$, all possible feasible splits are considered by filling in any one available slot in one child and restricting the corresponding slot in the other child; this ensures mutually exclusive order representations. The split with the optimal score improvement is made, and the tree growing procedure continues until the score cannot be improved. Data structure $\mathcal{I}$ stores information for visited nodes, since it is possible to re-use prior information at a later splitting decision. Computing the score for any order representation in the tree requires scanning the dataset and computing *ordinal summary statistics* (the counts and durations in Equation 1); this procedure is outlined in Appendix B.1 [Bhattacharjya, Gao, and Subramanian 2020]. The following result shows that the order in which nodes in $\mathcal{R}$ are visited does not matter in the greedy procedure.

Theorem 8. *The greedy optimal split decision at any node in the candidate list of nodes $\mathcal{R}$ during Algorithm 1 does not*

depend on decisions at other nodes.

In most real-world datasets, a sub-tree does not grow maximally towards the full tabular representation; in fact, the entire point is to avoid this situation through parameter sharing. However, if this does indeed occur, the worst case complexity for sub-tree construction is as follows:

Theorem 9. *The number of nodes and edges in the maximally grown sub-tree for order representation length k is $O(M^k)$, and the worst-case time complexity for sub-tree learning is upper bounded by $O(kKM^kN)$, assuming that event labels occur in roughly the same proportion. Here M is the number of event labels, N is the number of events and $K = |\mathbf{U}|$ is the parent set size.*

4.2 Learning Windows

Windows w_X for all nodes $X \in \mathcal{L}$ can be learned by maximizing the log likelihood over an event dataset. We begin by illustrating some challenges around optimizing the log likelihood to obtain windows, and then propose a heuristic approach for window estimation.

Formulation & Challenges. Let us consider a node X in a tree OGEM and study how one may learn window w_X that optimizes log likelihood; all observations extend to tabular OGEMs as well. When its parents $\mathbf{U}$ are known, recall that the log likelihood on an event dataset can be written in terms of counts and durations; after replacing maximum likelihood estimates for conditional intensities in Equation 1:

$$\text{logL}_X(\mathcal{D})^* = \sum_{\mathbf{r_L}} \left(-N(x; \mathbf{r_L}) + N(x; \mathbf{r_L}) \log \left(\frac{N(x; \mathbf{r_L})}{D(\mathbf{r_L})} \right) \right) \quad (3)$$

Both counts $N(x; \mathbf{r_L})$ and durations $D(\mathbf{r_L})$ are functions of the window w_X, making this a non-linear univariate optimization problem. The maximization problem is however not concave; in general, it has a large number of local maxima, depending on how the various counts and durations depend on window w_X. This is illustrated in Appendix B.2 using the simplest non-trivial case for order-dependent conditional intensities for any node X in an OGEM.

A Heuristic Algorithm. To avoid the afore-mentioned intractable optimization, we propose a heuristic that leverages the fact that when a node X has only one parent Z, both tabular and tree OGEMs are identical to a proximal GEM [Bhattacharjya, Subramanian, and Gao 2018], making it relatively easy to find the optimal window. We use the following notation: $\{\hat{t}_{zx}\}$ refers to the set of inter-event times from the most recent occurrence of Z, if Z has occurred, to every occurrence of X ($Z \neq X$), and $\{\hat{t}_{zz}\}$ denotes inter-event times between Z occurrences, including the time from the last occurrence of Z to the final time T.

Theorem 10. *For a node X with a single parent Z, the log likelihood maximizing window w_X either belongs to or is a left limit of a window in the candidate set $W^* = \{\hat{t}_{zx}\} \cup \max\{\hat{t}_{zz}\}$, where $\{\hat{t}\}$ denotes inter-event times.*

The above result shows that when a node has a single parent, one can discover a small number of local maxima from the inter-event times in the data, thereby easily computing the global maximum by exhaustively comparing all local maxima. Left limiting points are inspected though a hyper-parameter ϵ, chosen to be 0.001 for all experiments [Bhattacharjya, Subramanian, and Gao 2018].

Algorithm 2 Learn Window and Parameters

procedure LEARNWINDOW(event label X, parents $\mathbf{U}$, masking function $\phi(\cdot)$, dataset $\mathcal{D}$, type of model learner: 'tabular' or 'tree')
  Compute inter-event times $\{\hat{t}_{zx}\}$ and $\{\hat{t}_{zz}\}$ by scanning through the dataset in $\mathcal{D}$
  Compute candidate window pairs $\mathcal{W}^c = \{w_{ZX}\}$, $\forall Z, X \in \mathcal{L}$, assuming that Z is the only parent of X
  $S^* \leftarrow -Inf$
  for each label Z in $\mathbf{U}$ **do** ▷ Loop over all parents
    Compute all model information including $S_X(\mathbf{U})$ and Λ_X, using procedure 'ComputeScore' with window $w_X = w_{ZX}$ from $\mathcal{W}^c$
    if $S_X(\mathbf{U}) > S^*$ **then**
      $S^* \leftarrow S_X(\mathbf{U})$; $w_X^* \leftarrow w_{ZX}$; $\Lambda_X^* \leftarrow \Lambda_X$
  return w_X^*, Λ_X^*

procedure COMPUTESCORE(event label X, parent set $\mathbf{Pa}(\mathbf{X})$, window w_X, masking function $\phi(\cdot)$, dataset $\mathcal{D}$, type of model learner: 'tabular' or 'tree')
  if Learner type is 'tabular' **then**
    Compute ordinal summary statistics
    Compute max. likelihood parameter estimates, log likelihood at these estimates and score (Equation 2)
  else ▷ Learner type is 'tree'
    Run the procedure 'OptTree' from Algorithm 1
    Compute total score by summing the scores of leaf representations in $\mathcal{I}$
  return Λ_X and score $S_X(\mathbf{Pa}(\mathbf{X}))$

An approach for learning windows using the above result (as well as conditional intensity parameters), given the parent set $\mathbf{U}$, is outlined in Algorithm 2. This is shown for both tabular and tree versions of OGEM. Here, inter-event times are computed and then candidate windows are obtained by assuming that a parent Z is the *only* parent of X. For each candidate window – one for every parent of X – we compute the optimal model, and retain the model and window with the highest score. The OGEM-tree-W model from the experimental section later in the paper uses this heuristic procedure along with a tree learner for estimating model parameters.

4.3 Learning Parents

Since OGEMs fit within the broad framework of GEMs, we use a standard score-and-search algorithm to learn the parents for each node. Specifically, we use the Bayesian information criterion (BIC) score (Equation 2) with a forward-backward search algorithm, which has also been deployed for related graphical models [Bhattacharjya, Subramanian, and Gao 2018; Nodelman, Shelton, and Koller 2003]. At each iteration, the search algorithm greedily tests whether to add or remove a parental candidate for the target node by checking whether the BIC score improves with the modified

parental set. The algorithm is outlined in Appendix B.3.

Theorem 11. *A forward-backward parent search approach for node X with known window w_X and a constant maximum number of parents, using tree learning from Algorithm 1, has worst-case time complexity polynomial in the number of event labels M and linear in the number of events N.*

5 Experiments

5.1 Model Fitting

We test the proposed models on the task of model fitting.

Datasets. We consider the following datasets (see the second and third columns of Table 1 for data size information):

- ICEWS [O'Brien 2010]: These are single stream datasets involving socio-political events, one each for 4 countries in the Integrated Crisis and Early Warning System (ICEWS) political event dataset. We chose 4 out of 5 Latin American countries from the ICEWS extract in Bhattacharjya *et al*. [2018]. Venezuela was omitted due to inconsistency between labels while splitting the data into three sets.
- IPTV[2] [Luo et al. 2015]: This dataset records users' TV watching behavior over almost a year, where event labels are the TV program categories. We sample 100 users.
- LinkedIn [Xu, Luo, and Zha 2017]: This dataset includes employment related new roles in a company for anonymous LinkedIn users. We sample 1000 employees.
- Mimic-II [Saeed et al. 2011]: These are patient electronic health records from Intensive Care Unit visits over 7 years. Small sequences are filtered out, resulting in 650 patients.
- Stack Overflow[3] [Grant and Betts 2013]: This tracks user behavior around receipt of badges to encourage engagement in a question answering website. We sample 1000 users from the Du *et al.* [Du et al. 2016] data who acquire one or more of 20 types of badges.

Models. The OGEMs with tabular and tree representations are referred to as OGEM-tab and OGEM-tree respectively. In these models, the user is required to specify a candidate set of windows as hyper-parameters. OGEM-tree-W refers to the tree OGEM where windows are learned automatically, without user input, using the heuristic in Section 4.2. Two other parametric GEM baselines are also compared:

- Multivariate Hawkes process (MHP) [Bacry et al. 2017]: We consider a classic multivariate Hawkes process model with exponential kernels that involves a decay rate and an infectivity matrix. We use the *tick*[4] library which takes a least squares objective for learning; this is efficient and results in a convex quadratic programming problem when the decay rate is fixed. (See Appendix C.1 for details.)
- Proximal graphical event model (PGEM) [Bhattacharjya, Subramanian, and Gao 2018]: This piece-wise constant model allows different windows for different parents but only accounts for whether a parent has occurred within its proximal window. Importantly, it does not distinguish between orders of causal events. Here the learning approach also identifies windows using a heuristic.

Experimental Setup. Each dataset is partitioned into three splits: train (70%), dev (15%) and test (15%), only retaining common event labels across these splits. We use time to achieve the splits in single stream datasets like ICEWS countries, e.g. if $T = 100$ days, then events up to time 70 days are in train. We use the user id to split multiple stream datasets like IPTV, e.g. for 100 users, streams for 70 of them make up the train set. We measure each model's performance by the log likelihood on the held-out test set.

For all models, if there is a parametric condition that is not observed in the training data, we use a default intensity rate, denoted λ^0, treated as a hyper-parameter; this is particularly important for OGEM-tab and a major limitation, since it may be possible for some order instantiations to never be observed during training. All hyper-parameter choices are listed in Appendix C.1. Runtimes for all models range from minutes to hours depending on the dataset size; these can be reduced with optimized code and parellization.

Results. Table 1 shows the model goodness-of-fit evaluated by the log-likelihood on the held-out test sets for the 4 single stream ICEWS datasets as well as the 4 multiple stream datasets. For all OGEM based models, the better performing result between masking function 'first' vs. 'last' is shown; we note that this choice does not have much impact for the datasets considered. The results highlight that with the exception of IPTV and the Mexico ICEWS dataset, the proposed ordinal models that exploit order tree representations (OGEM-tree and OGEM-tree-W) outperform all other models. The IPTV dataset in particular is clearly better suited for self-exciting patterns from a Hawkes process, which explains its popularity in this literature. Since people are likely to continue to watch the same kinds of shows in a short time period, an OGEM with its masking function that reduces a sequence of the same label to a singleton is perhaps inappropriate here. For the other datasets, however, tree OGEMs prove adept at learning distinctive rate parameters for select order instantiations involving several parents. While OGEM-tab could do the same in principle, i.e. discover ordinal dependence involving several parents, it incurs a high penalty due to the resulting parametric complexity. OGEM-tree and OGEM-tree-W exploit the proposed order representation to economize on parametric complexity across several order instantiations over larger parental sets.

5.2 Case Studies

The primary benefit of OGEMs, particularly those with the tree representation, is that they can be an important component of an analyst's toolkit during the process of knowledge discovery from event stream data. They are helpful for gleaning information and better understanding temporal relationships between event labels. The tree-based representation allows learning more causal parents from limited data. We briefly demonstrate this with illustrative examples.

[2] https://github.com/HongtengXu/Hawkes-Process-Toolkit/tree/master/Data

[3] https://archive.org/details/stackexchange

[4] https://x-datainitiative.github.io/tick/modules/hawkes.html

Dataset	N (# events)	M (# labels)	MHP	PGEM	OGEM-tab	OGEM-tree	OGEM-tree-W
ICEWS							
Argentina	3252	104	-1419	-1386	-1369	**-1366**	-1393
Brazil	4249	114	-2169	-2000	-2057	-2050	**-1993**
Colombia	841	79	-528	-534	**-518**	**-518**	-537
Mexico	1905	97	**-760**	-797	-771	-769	-766
IPTV	332980	16	**-64168**	-77009	-75114	-72696	-74491
LinkedIn	2932	10	-1593	-1462	-1478	-1418	**-1406**
Mimic	2419	75	-567	-500	-474	**-429**	-454
Stack Overflow	71254	22	-52543	-48323	-49344	-49192	**-48232**

Table 1: Dataset information and log likelihood results for the models on the test sets.

MHP

Parents:
[2, 3, 7, 8, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20]

PGEM

Parents: [2, 4, 6]

Condition	Intensity Rate
ϕ	0.005
Only 2	0.021
Only 4	0.003
Only 6	0.028
2 & 4	0.017
4 & 6	0.016
2 & 6	0.047
2, 4 & 6	0.029

OGEM-tab

Parents: [2, 6]

Condition	Intensity Rate
ϕ	0.004
[2]	0.02
[6]	0.024
[6,2]	0.039
[2,6]	0.048

OGEM-tree

Parents: [2, 3, 4, 6, 14]

Condition	Intensity Rate
ϕ	0.005
[4]	0.003
[¬4]	0.023
[4, ¬4]	0.014
[¬4,4]	0.015
[¬4,¬4]	0.035
[?,?,?]	0.034
[?,?,?,?]	0.036
[?,?,?,?,?]	0.048

Figure 2: Parents and conditional intensity parameters across some models for badge $\#6$ of Stack Overflow.

Stack Overflow. Figure 2 compares the learned parents and some parameters from various models on a particular label – badge $\#6$ – from the Stack Overflow dataset. Here MHP learns a dense infectivity matrix and therefore many parents, making it less useful for understanding causal factors (parameters for MHP as well as experimental details are supplied in Appendix C.2). While it is possible to learn a sparser and more interpretable infectivity matrix in MHP, the model only allows excitation and cannot account for inhibitory effects. OGEM-tab only learns few parents due to the large penalty it incurs from model complexity. Both PGEM and OGEM-tree show that badge $\#4$ can have an inhibitory effect on acquiring badge $\#6$. OGEM-tree also recognizes that someone with more distinct kinds of badges is more likely to receive this badge; it can do so because it only uses 9 parameters over 5 parents in this case.

ICEWS. OGEMs could also be suitable while mining a dataset for finding parental orders associated with the highest conditional intensity rates. Studying these orders could help an analyst synthesize 'stories' around event labels of interest. From analysis on a larger extract of a few Latin American countries in the ICEWS political event dataset, where events are tuples of the form (Source Actor, Action, Target Actor), we observe historical order often has a major effect on events involving protest or fighting between actors. In particular, an escalation of events or a conciliatory effort immediately followed by aggression increases the rate of protests. We share two examples of this analysis, shown as parental order instantiation with the highest rate $\longrightarrow$ event label of interest:

- (Citizen (Peru), Appeal, Head of Govt. (Peru)); (Police (Peru), Fight, Citizen (Peru)) $\longrightarrow$ (Protester (Peru), Protest, Govt. (Peru))
- (Protester (Venezuela), Protest, Military (Venezuela)); (Military (Venezuela), Fight, Citizen (Venezuela)); (Protester (Venezuela), Protest, Govt. (Venezuela)) $\longrightarrow$ (Protester (Venezuela), Protest, Govt. (Venezuela))

For this analysis, we used the 'last' masking function $\phi(\cdot)$, $\gamma = 0.1$ and $\lambda^0 = 0.001$.

6 Conclusions

We have proposed a novel graphical event model that uses a tree representation for capturing ordinal historical dependence in conditional intensity functions for multivariate point processes. Specifically, we make the following contributions: 1) an efficient tree ordinal GEM, improving upon a simpler tabular representation; 2) practical efficient learning algorithms of tree OGEM parameters as well as structure, plus a heuristic for estimating windows that could also be applied to prior work; and 3) empirical studies that show the competitive performance of tree OGEM over relevant baselines. Future work could pursue other heuristics for learning windows such as those suggesting a more comprehensive set of candidate windows, as well as variations involving choice of masking function and other compact parameter representations.

Acknowledgments

The authors thank the anonymous reviewers for their helpful feedback and suggestions. This research was developed with funding from the Defense Advanced Research Projects Agency (DARPA). The views, opinions and/or findings expressed are those of the authors and should not be interpreted as representing the official views or policies of the Department of Defense or the U.S. Government.

Ethical Statement

This paper is concerned with a particular class of graphical event model, namely a tree-based ordinal graphical event model (OGEM). Our contribution here is primarily around formulating the model and proposing a machine learning approach that uses event stream data involving various kinds of events. The applicability is therefore broad, making the model suitable for all sorts of domains with access to such datasets, including system reliability, social networks, manufacturing processes, retail, healthcare, politics and finance. Since the proposed model is able to explicitly capture ordinal historical dependence from an event label's parent causes, we have made the case that it could be a useful addition to an analyst's toolkit around discovery from event datasets.

One has to exercise necessary caution in interpreting the graphical event model structure resulting from our proposed model and algorithm as a causal graph. While the OGEM tree model has a specific causal semantic interpretation with respect to the dependence of the instantaneous rate of a child node on the order of historical event occurrences of event types in its parental set, it might be better understood as a statistical, multivariate stochastic process model, particularly since latent variables are often prevalent in real-world datasets. Our work here is observational and not based on interventions for deducing controlled cause-effect relationships, which is the gold standard in causal inference. Practitioners that apply these models in real-life problems should therefore exercise necessary caution with respect to drawing any cause-effect conclusions based on the resulting models. Juxtaposing relevant domain knowledge and expertise in the chosen application area alongside our proposed model would indeed be beneficial towards interpreting the accompanying cause-effect semantics carefully and appropriately.

The proposed learning algorithm is dependent on the event dataset, so we note that any bias in the sensing and recording of events in the input dataset naturally carries over into the learned model structure as well as parameters that result from our algorithm. However, as long as the data collection process is appropriately set up, the authors do not see any bias or fairness related concern stemming from the proposed innovation that exploits order representations to express and learn an OGEM tree model.

References

Aalen, O. O.; Borgan, O.; and Gjessing, H. K. 2008. *Survival and Event History Analysis: A Process Point of View*. New York, NY, USA: Springer Science & Business Media.

Bacry, E.; Bompaire, M.; Deegan, P.; Gaïffas, S.; and Poulsen, S. V. 2017. Tick: A Python library for statistical learning, with an emphasis on Hawkes processes and time-dependent models. *Journal of Machine Learning Research* 18(1): 7937–7941.

Bacry, E.; Mastromatteo, I.; and Muzy, J.-F. 2015. Hawkes processes in finance. *Market Microstructure and Liquidity* 1(01): 1550005.

Bhattacharjya, D.; Gao, T.; and Subramanian, D. 2020. Order-dependent event models for agent interactions. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 1977–1983.

Bhattacharjya, D.; Subramanian, D.; and Gao, T. 2018. Proximal graphical event models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 8147–8156.

Boutilier, C.; Friedman, N.; Goldszmidt, M.; and Koller, D. 1996. Context-specific independence in Bayesian networks. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI)*, 115–123.

Cox, D. R.; and Lewis, P. A. W. 1972. Multivariate point processes. In *Proceedings of the Sixth Berkeley Symposium on Mathematical Statistics and Probability, Volume 3: Probability Theory*, 401–448.

Didelez, V. 2008. Graphical models for marked point processes based on local independence. *Journal of the Royal Statistical Society, Ser. B* 70(1): 245–264.

Du, N.; Dai, H.; Trivedi, R.; Upadhyay, U.; Gomez-Rodriguez, M.; and Song, L. 2016. Recurrent marked temporal point processes: Embedding event history to vector. In *Proceedings of the SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 1555–1564.

Etesami, J.; Kiyavash, N.; Zhang, K.; and Singhal, K. 2016. Learning network of multivariate Hawkes Processes: A time series approach. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI)*, 162–171.

Gao, T.; Subramanian, D.; Shanmugam, K.; Bhattacharjya, D.; and Mattei, N. 2020. A multi-channel neural graphical event model with negative evidence. In *Proceedings of the Conference on Artificial Intelligence (AAAI)*, 3946–3953.

Geiger, D.; and Heckerman, D. 1996. Knowledge representation and inference in similarity networks and Bayesian multinets. *Artificial Intelligence* 82(1-2): 45–74.

Grant, S.; and Betts, B. 2013. Encouraging user behaviour with achievements: An empirical study. In *Proceedings of the IEEE Working Conference on Mining Software Repositories (MSR)*, 65–68.

Gunawardana, A.; Meek, C.; and Xu, P. 2011. A model for temporal dependencies in event streams. In *Advances in Neural Information Processing Systems (NeurIPS)*, 1962–1970.

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

Luo, D.; Xu, H.; Zhen, Y.; Ning, X.; Zha, H.; Yang, X.; and Zhang, W. 2015. Multi-task multi-dimensional Hawkes processes for modeling event sequences. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 3685–3691.

Meek, C. 2014. Toward learning graphical and causal process models. In *Proceedings of the Uncertainty in Artificial Intelligence Workshop on Causal Inference: Learning and Prediction*, 43–48.

Mei, H.; and Eisner, J. M. 2017. The neural Hawkes process: A neurally self-modulating multivariate point process. In *Advances in Neural Information Processing Systems (NeurIPS)*, 6754–6764.

Nodelman, U.; Shelton, C. R.; and Koller, D. 2003. Learning continuous time Bayesian networks. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI)*, 451–458.

O'Brien, S. P. 2010. Crisis early warning and decision support: Contemporary approaches and thoughts on future research. *International Studies Review* 12: 87–104.

Parikh, A. P.; Gunawardana, A.; and Meek, C. 2012. Conjoint modeling of temporal dependencies in event streams. In *Proceedings of the Uncertainty in Artificial Intelligence Workshop on Bayesian Modeling Applications*.

Pearl, J. 1988. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Poole, D.; and Zhang, N. L. 2003. Exploiting contextual independence in probabilistic inference. *Journal of Artificial Intelligence Research* 18: 263–313.

Saeed, M.; Villarroel, M.; Reisner, A. T.; Clifford, G.; Lehman, L.-W.; Moody, G.; Heldt, T.; Kyaw, T. H.; Moody, B.; and Mark, R. G. 2011. Multiparameter intelligent monitoring in intensive care II (MIMIC-II): A public-access intensive care unit database. *Critical Care Medicine* 39(5): 952.

Simma, A.; Goldszmidt, M.; MacCormick, J.; Barham, P.; Black, R.; Isaacs, R.; and Mortie, R. 2008. CT-NOR: Representing and reasoning about events in continuous time. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI)*, 484–493.

Xiao, S.; Yan, J.; Yang, X.; Zha, H.; and Chu, S. M. 2017. Modeling the intensity function of point process via recurrent neural networks. In *Proceedings of the Conference on Artificial Intelligence (AAAI)*, volume 17, 1597–1603.

Xu, H.; Luo, D.; and Zha, H. 2017. Learning Hawkes processes from short doubly-censored event sequences. In *Proceedings of the International Conference on Machine Learning (ICML)*, 3831–3840.

Yu, X.; Shanmugam, K.; Bhattacharjya, D.; Gao, T.; Subramanian, D.; and Xue, L. 2020. Hawkesian graphical event models. In *Proceedings of the International Conference on Probabilistic Graphical Models (PGM)*.