EXPLAINING POINT PROCESSES BY LEARNING INTERPRETABLE TEMPORAL LOGIC RULES

Anonymous authors
Paper under double-blind review

ABSTRACT

We propose a principled method to learn a set of human-readable logic rules to explain temporal point processes. We assume that the generative mechanisms underlying the temporal point processes are governed by a set of first-order temporal logic rules, as a compact representation of domain knowledge. Our method formulates the rule discovery process from noisy event data as a maximum likelihood problem, and designs an efficient and tractable branch-and-price algorithm to progressively search for new rules and expand existing rules. The proposed algorithm alternates between the rule generation stage and the rule evaluation stage, and uncovers the most important collection of logic rules within a fixed time limit for both synthetic and real event data. In a real healthcare application, we also had human experts (i.e., doctors) verify the learned temporal logic rules and provide further improvements. These expert-revised interpretable rules lead to a point process model which outperforms previous state-of-the-arts for symptom prediction, both in their occurrence times and types.

1 INTRODUCTION

Event sequences with irregular time intervals are ubiquitous. The inter-event times usually convey rich information regarding the underlying dynamics such as disease progression (Liu et al., 2015). It is useful to understand events' generating mechanisms, as well as the occurrence reason and time. In systems where domain knowledge is rich, events generating can usually be governed by a few *first-order temporal logic rules*, as a compact representation of knowledge. In healthcare, the knowledge "if a sudden fall in blood pressure is observed, vasopressors are required to be applied to patients immediately; and then the blood pressure may return to normal afterwards" may explain why the event "blood pressure returns to normal from the abnormally low level " is observed after the event "vasopressors are used". This domain knowledge can be summarized as a collection of logic rules with *temporal relation constraints*, such as "A happens *before* B", "If A happens, and *after* 5 mins, B can happen", and "If A and B happen *simultaneously*, then *at the same time* C can happen". However, temporal logic alone is not an ideal temporal model, as hard constraints will be too strict to model the recurrent noisy event data.

Meanwhile, a large amount of literature has been devoted to modeling event data, among which *temporal point process* (TPP) models provide an elegant framework without the need to discretize the time horizon into bins and to compute the count of events within each bin. TPP models treat the inter-event time as random variables and directly model the intensity function (i.e., occurrence rate) of the events. However, most TPP models lack interpretability, and they can not represent domain knowledge in a human-readable form. Recently, Li et al. (2020) proposed a unified framework to marry *point process intensity functions* with *temporal logic rules*. This method employs a set of pre-specified temporal logic rules to design the intensity functions to incorporate domain knowledge. The resulting so-called *temporal logic point process* models are inherently interpretable and expressive. Using such logic-informed intensity functions, the model can capture nonlinear dependencies, interactions, and various temporal relations between events. It was also shown that, by specifying one or two simple temporal logic rules, many existing parametric models, e.g. the Hawkes process (i.e. self-exciting processes) (Hawkes, 1971a) and the self-correcting process (Isham & Westcott, 1979) are special cases of the proposed temporal logic point process.

However, in (Li et al., 2020) the temporal logic rules are required to be specified by the model builder, but they may not be known beforehand. Models constructed in such a way may suffer from model misspecification when the pre-specified logic rules are incorrect or missing.

Can we automatically discover the temporal logic rules governing the event dynamics based on historical event data alone?

This is a challenging problem since the space of possible temporal logic formulas is huge, comprising of a combination of a large number of discrete logic variables and all kinds of temporal relations.

Previous inductive logic programming approaches focused mostly on deriving the relations between discrete logic variables, but have largely ignored the crucial temporal information (Dash et al., 2018; Wei et al., 2019). A generalization of these methods by further considering temporal relations in logic variables is largely missing and is in pressing needs due to the increasing availability of event data. To address the aforementioned challenges, we propose the TEmporal Logic rule LearnER (TELLER) algorithm for learning temporal logic rules from event sequences. TELLER is inspired by the branch-and-price algorithm (Barnhart et al., 1998), a column generation (CG) algorithm for linear programming (LP) problems where the number of variables is too large to be considered explicitly. In the CG algorithm framework, the original LP problem is solved via two alternating procedures: the master problem and the subproblem, where the master problem is the original problem with only a subset of variables being considered and the subproblem is a new problem created to identify a new variable to be added. Similarly, TELLER also alternates between solving a master problem and a subproblem, where the master problem aims to evaluate the current rules by maximizing the likelihood and reweighting these rules as in (Li et al., 2020), and the subproblem is to search and construct a new temporal logic rule (by extending a current rule or adding a new short rule). It repeats until the likelihood can no longer be improved by adding new candidate rules. In this way, TELLER searches through the vast space of potential temporal logic rules and learns the importance weights of the discovered rules to hedge against noise in the data. Fig. 1 shows the flow.

Specifically, TELLER is computationally efficient and with guarantees. It includes the candidate rules one by one in a tractable and progressive fashion, and is able to find a near-optimal set of rules. The hypothesized logic rules are in disjunctive normal form (DNF, OR-of-ANDs) with temporal relation constraints, and can be of various lengths. Given the temporal logic point process modeling framework, we show that the resulting objective function, i.e. likelihood, is convex, which guarantees optimal performance for our search algorithm. A set of logic rules and their importance weights are jointly learned by maximizing the likelihood. The most important collection of logic rules are guaranteed to be discovered within a fixed time limit.

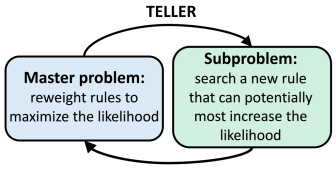

Figure 1: The flow of TELLER. It alternates between a rule evaluation stage (master problem) and a rule proposal stage (subproblem).

The uncovered rules by TELLER will not only shed light on *when* and *what* events would happen, but also *why* some events would happen at a specific time. Our method facilitates the human-readable knowledge exchange between experts and point process models. On one hand, the mined temporal logic rules may supplement or refine the existing knowledge; on the other hand, human experts can easily provide feedback to modify the learned models via logic rules, as a way to add the safety of using these models in high-stake tasks like healthcare and autonomous driving. In our real experiment related to medicine, TELLER is used to learn the explanatory temporal logic rules regarding the choice and arrangement of drugs in improving patients' health status. Doctors were asked to justify the rules and they confirmed that these discovered logic rules are consistent with the pathogenesis and have captured the most important factors in affecting patients' health status.

2 Background

2.1 First-order Temporal Logic Rules

First-order temporal logic is a form of symbolized reasoning in which each statement is a composition of temporal predicates and their relations. We formally define the interval-based temporal logic below. This type of temporal logic rule is often suitable for reasoning about events with duration, which are better modeled if the underlying temporal ontology uses time intervals. It fits well with the temporal point process models for event sequences, where the time intervals of events are explicitly modeled. Refer to (Goranko & Rumberg, 2021) for a comprehensive survey of various temporal logic models.

Temporal predicate. First define a set of *static predicates* $\mathcal{X} = \{X_1, \ldots, X_d\}$, where each static predicate $X_i(c)$ is a logic random variable that defines the *property* or *relation* of entities, such as Smokes(c) or Friend(c, c'), where c and c' are the entities. By adding a temporal dimension to predicate X_i, we obtain a *temporal predicate* $\{X_i(c, t)\}_{t \geq 0}$, which can be viewed as a continuous-time stochastic process. Given observations, each temporal predicate will be grounded as a list of ordered

0-1 events, which take True (1) and False (0) in an alternating way with the state transition times recorded. For example, a temporal predicate $\{\text{NormalBloodPressure}(c,t)\}_{t\geq 0}$, where entity c is referring to the patient, will take value 1 or 0 at any time t to indicate whether the patient's blood pressure is normal or not, with stochastic state transition time (see Fig. 2).

To simplify the notation, we will temporally drop the dependency of predicates on entities. We will write $X(c,t)$ as $X(t)$ instead and the grounded temporal predicate as $\{x(t)\}_{t\geq 0}$.

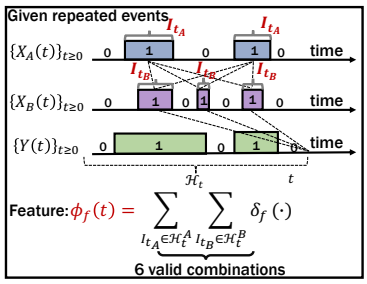

Figure 2: Feature constructions of TLPP using a simple logic formula $f: Y \leftarrow A \wedge B$, as a template to gather combinations of the body predicate history events. Here, A has 2 events and B has 3 events, leading to 6 valid combinations.

2.2 Temporal Logic Point Process

Temporal Point Process (TPP) provides an elegant tool to capture the dynamics of the event sequences. It is characterized by conditional intensity function, denoted by $\lambda(t|\mathcal{H}_t)$, where history $\mathcal{H}_t$ is the knowledge of times of all events. By definition, we have $\lambda(t|\mathcal{H}_t)dt = \mathbb{E}[N([t,t+dt])|\mathcal{H}_t]$, where $N([t,t+dt])$ denotes the number of points falling in an interval $[t,t+dt]$. Here we aim to use TPP to model these 0-1 events and use the intensity function to capture the 0-1 transition rate.

Temporal relation. We will use the *interval-based temporal relations* originally introduced in Allen's seminal paper (Allen, 1990) to describe the temporal relations that can exist in any two temporal predicates. There are 13 types of possible temporal relations, including *Before, Meets, Overlap* and so on (see Appendix B for a comprehensive illustration), and we denote this set by $\mathcal{R}$. To evaluate these temporal relations, we need to define the *time interval* using the state transit in/out times. For example, at time t_A, the temporal predicate $\{X_A(t)\}_{t\geq 0}$ takes state $x_A = 0$ or 1, we define $I_{t_A} = (t_{A_1}, t_{A_2}]$, $t_A \in (t_{A_1}, t_{A_2}]$, as a time interval where t_{A_1} is the transition time that this predicate enters the state x_A and t_{A_2} is the transition time that this predicate leaves the state x_A. For any two temporal predicates, given their time intervals, $I_{t_A} = (t_{A_1}, t_{A_2}]$ and $I_{t_B} = (t_{B_1}, t_{B_2}]$, their temporal relation function, denoted as $R_{A\ type\ B}(I_{t_A}, I_{t_B}) \in \mathcal{R}$, is a logic function defined over the time intervals and can be evaluated by plugging in the specific time intervals. For example, $R_{A\ Before\ B} = \mathbb{1}\{t_{B_1} > t_{A_2}\}$ and $R_{A\ Equals\ B} = \mathbb{1}\{t_{A_1} = t_{B_1}\}\mathbb{1}\{t_{A_2} = t_{B_2}\}$. All these definitions of temporal relations are also given in Appendix B.

Temporal logic formula. A first-order temporal logic rule f is defined as logical connectives of *temporal predicates* and their *temporal relations*. The generic form is:

$$f: (\neg)Y(t_y) \leftarrow \bigwedge_{X_u \in \mathcal{X}_f^+} X_u(t_u) \bigwedge_{X_v \in \mathcal{X}_f^-} \neg X_v(t_v) \bigwedge_{X_u, X_v \in \mathcal{X}_f} R_{u\ type\ v}(I_{t_u}, I_{t_v}) \tag{1}$$

where each temporal predicate and the temporal relations can be evaluated at t_y, t_u and t_v along with their corresponding intervals I_{t_u} and I_{t_v}. In the above formula, negation $(\neg)$ means the head predicate Y can take a negative sign; $\mathcal{X}_f = \mathcal{X}_f^+ \cup \mathcal{X}_f^-$ is the set of predicates defined in f, where $\mathcal{X}_f^-$ is the set of predicates as negation in the formula f, and $\mathcal{X}_f^+ = \mathcal{X}_f \setminus \mathcal{X}_f^-$. We assume $\leftarrow$ has a causal direction, and the body part of the formula indicate the evidence to be gathered from history to deduce the state of the head predicate. By this assumption, it is only valid to consider $t_y \geq t_u, t_v$.

Given observed sequences of data, each predicate in $\mathcal{X}$ will be grounded as a list of ordered 0-1 events. The main idea of *temporal logic point process* (TLPP) (Li et al., 2020) is to use the temporal logic rules (1) as templates to selectively choose *combinations of events* from history as evidence to infer the transition rate of these 0-1 events. In this way, the structure of the intensity function will be informed by the pre-specified temporal logic rules. Below we provide more details.

Logic-informed intensity function. Let $\mathcal{H}_t$ include all the historical trajectories of the grounded predicates in $\mathcal{X}$ up to t. TLPP leverages the rule set to model the state transition dynamics of the head predicates. Define $\lambda^*(t) := \lambda(t|\mathcal{H}_t)$ as the transition intensity for $\{Y(t)\}_{t\geq 0}$ to transit from 0 to 1, given history up to t; and $\mu^*(t) := \mu(t|\mathcal{H}_t)$ vice versa. Assume the generating process of the 0-1 events are governed by the set of temporal logic rules $\mathcal{F} = \{f_1, f_2, \dots\}$.

First consider one formula f, to incorporate its knowledge in intensity construction, we define a *formula effect* (FE), which aims to gather only the effective combinations of the historical predicate events defined by f as evidence to reason about the transition rate of the head predicate. To ease the notation, define the predicate index set of $\mathcal{X}_f$ as U, then the *formula effect* of f is computed as

$$\delta_f\left(t \mid y(t), \{x_u(t_u), I_{t_u}\}_{u\in U} \in \mathcal{H}_t\right) := f\left(1 - y(t), \{x_u(t_u), I_{t_u}\}_{u\in U}\right) - f\left(y(t), \{x_u(t_u), I_{t_u}\}_{u\in U}\right) \tag{2}$$

where $y(t)$ is the observed head predicate state at t, $1 - y(t)$ is its counterfactual state, $\{x_u(t_u), I_{t_u}\}_{u\in U} \in \mathcal{H}_t$ is one historical combination of the body predicates (including their states

and time intervals) in f, and $f(\cdot)$ is the clausal form of (1), which is an alternative expression to ease the evaluation of the rule. For example, $Y \leftarrow X$ is logically equivalent to its clausal form $\neg Y \vee X$. FE, which is essentially a difference between the what-if scenario and the true scenario, answers the question such as "should $Y(t)$ transit its state given logic formula f is true".

One can check that the sign of FE can only be 1, -1 or 0, which can be interpreted as: $\text{sign}(\text{FE}) = 1$ indicates a positive effect for the head to transit, -1 indicates a negative effect, and 0 means no effect. Only the non-zero FE will refer to an effective combination. Let $\mathcal{H}_t^u$ be the historical trajectory specific to predicate u up to t, and one can aggregate all the valid (i.e., non-zero) FEs from history by a summation over all combinations of the temporal predicate states and their intervals, i.e.,

$$\phi_f(t) = \sum\nolimits_{\{(x_u(t_u), I_{t_u}) \in \mathcal{H}_t^u\}_{u \in U}} \delta_f(t \mid y(t), \{x_u(t_u), I_{t_u}\}_{u \in U}), \tag{3}$$

where $\phi_f(t)$ is the *feature* informed by logic formula f. One can refer to Fig. 2 for an illustration.

For each $f \in \mathcal{F}$, one can build the rule-informed and history-dependent features $\phi_f(t)$ as above, and assumes the rules are connected in disjunctive normal form (OR-of-ANDs) to deduce Y. Then the transition of $\{Y(t)\}_{t \geq 0}$ are modeled as monotonically increasing and non-negative functions of the weighted sum of the features:

$$\lambda^*(t) = \exp\left(b_0 + \sum\nolimits_{f \in \mathcal{F}} w_f \cdot \phi_f(t)\right), \qquad \mu^*(t) = \exp\left(b_1 + \sum\nolimits_{f \in \mathcal{F}} w_f \cdot \phi_f(t)\right) \tag{4}$$

where $\boldsymbol{w} = [w_f]_{f \in \mathcal{F}} \geq 0$ are the weight parameters associated with each rule $f \in \mathcal{F}$, and b_0, b_1 are the spontaneous intensity, which are distinct for the dual intensities. Weight parameters will be shared by the dual intensities, but the calculated features ϕ_f through (2) and (3) will always have opposite signs. One can think of the formula weight w_f as the confidence level put on f. The higher the weight, the more influence that the formula will have on the intensity. The rationale of TLPP is that the intensity functions are constructed in a way that the yielding 0-1 event sequences will enable the set of logic rules to be more satisfied.

3 The Proposed Method: TELLER

Likelihood function and its convexity. Given $\{\mathcal{H}_t\}_{0<t<T}$, one can write out the likelihood w.r.t. the intensity $\lambda^*(t)$ and $\mu^*(t)$. Suppose a realization of $\{Y(t)\}_{0<t<T} = \{Y(0) = 0, Y(t_1) = 1, \ldots, Y(t_n) = 1, t_n < T < t_{n+1}\}$, the (log) likelihood are (see proof in Appendix C): $\ell(\boldsymbol{w}, b_0, b_1) = \log \mathcal{L}(\boldsymbol{w}, b_0, b_1)$, where:

$$\begin{aligned}\mathcal{L}(\boldsymbol{w}, b_0, b_1) =& \lambda^*(t_1) \exp\left(-\int_0^{t_1} \lambda^*(s) ds\right) \cdot \mu^*(t_2) \\ & \exp\left(-\int_{t_1}^{t_2} \mu^*(s) ds\right) \cdots \exp\left(-\int_{t_n}^{T} \mu^*(s) ds\right),\end{aligned} \tag{5}$$

Note that $-\ell$ is convex w.r.t $\boldsymbol{w}$, b_0 and b_1. This is true as the intensity function has a functional form of Eq. (4), and this turns the log-likelihood function into a generalized linear model (GLM) with Poisson observations and log link (Fahrmeir & Tutz, 2013). It is well-known that the negative log-likelihood of a GLM is convex w.r.t the model parameters. This convexity property leads to a convergence guarantee for TELLER.

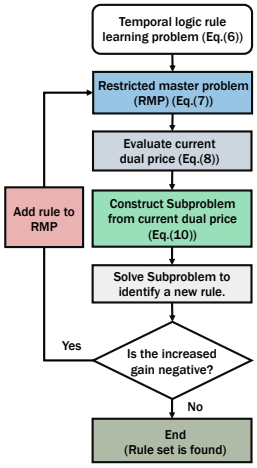

Figure 3: TELLER

We will design our TELLER with the backgrounds detailed in the previous section. TELLER uses TLPP as the backbone to evaluate the likelihood of the grounded 0-1 temporal predicate sequences. The likelihood will be maximized to jointly learn the set of temporal logic formulas and their weights. The problem is essentially combinatorial and requires enumerating an exponentially large set of combinations of the predicates and their signs. Our problem is even more challenging compared to traditional inductive logic programming due to the sequential properties of data with temporal information. TELLER solves this problem using the procedures as shown in Fig. 3. We will start our exposition with the formulation of the original, or the master rule learning problem.

3.1 Temporal Logic Rule Learning Problem (Master Problem)

We aim to uncover the set of temporal logic rules $\mathcal{F}$ based via optimizing:

$$\text{P}_{\text{Master}}: \quad \boldsymbol{w}^*, b_0^*, b_1^* = \underset{\boldsymbol{w}, b_0, b_1}{\text{argmin}} -\ell(\boldsymbol{w}, b_0, b_1) + \lambda_0 \sum\nolimits_{f \in \bar{\mathcal{F}}} c_f w_f; \quad s.t. \quad w_f \geq 0, \quad f \in \bar{\mathcal{F}} \tag{6}$$

where $\bar{\mathcal{F}}$ is the complete and exponentially large collection of *all possible* temporal logic formulas that can be created by the set of the pre-specified temporal predicates and their temporal relations. The algorithm needs to determine: 1) the assignment $\mathcal{X}_f$, $\mathcal{X}_f^+$, and $\mathcal{X}_f^-$; 2) the important temporal relations defined over the involved predicates in f; and 3) the weight parameter w_f as in (4). The

regularization penalties related to the rule complexity are incorporated into the objective to force sparsity and to trade off accuracy against rule simplicity. Rule complexity coefficient c_f can be the rule length if we want to learn both the sparse and short rules. Any affine function e.g. $\lambda_0 \sum_{f \in \bar{\mathcal{F}}} c_f w_f$ with $c_f \geq 0$, $\lambda_0 \geq 0$ should work here and will not change the convexity. One can further tune λ_0 to balance the negative likelihood and the complexity penalty via cross-validation. The sparsity of $\boldsymbol{w}^*$ will indicate the collection of rules learned and these non-zero weight rules will compose $\mathcal{F}$.

Key Idea: Although we can write the rule learning problem as a regularized convex optimization above, the set of variables (w_f) is exponentially large and can not be optimized simultaneously in a tractable way. The idea of TELLER is to start from a restricted master problem (RMP), where the search space is smaller and the solution is tractable but suboptimal. TELLER will gradually improve this solution by iteratively expanding the search space until the solution approximates the optimum. The idea is inspired by the fact that most of the candidate rules in $\bar{\mathcal{F}}$ will not be in $\mathcal{F}$ and will lead to a zero value weight in the optimal solution. We do not want to waste resources by including these redundant rules when we expand the search space to optimize (6). TELLER will always generate the rule that has the potential to improve the objective function, which is an any-time algorithm with the guarantee that the most important rules have been considered.

Our progressive algorithm gives rise to two questions: **1)** the criterion to determine the new rule to add; **2)** How to construct new candidate temporal logic rules, as will be tackled in the following.

3.2 Criterion to Add Rules

Restricted master problem (RMP). TELLER first replaces the original rule set $\bar{\mathcal{F}}$ by a small subset $\mathcal{F}_0 \subset \bar{\mathcal{F}}$ and gradually expands this subset to produce a nested sequence of subsets $\mathcal{F}_0 \subset \mathcal{F}_1 \subset \cdots \subset \mathcal{F}_k \subset \cdots$. It does so by adding candidate rules identified by subproblems (we will elaborate on this later). Note that the initial rule set $\mathcal{F}_0$ can be an empty set or any pre-defined small set.

For each $\mathcal{F}_k$, $k = 0, 1, \ldots$, TELLER solves the restricted master problem by replacing $\bar{\mathcal{F}}$ with $\mathcal{F}_k$:

$$\text{P}_{\text{Restricted}}: \ \boldsymbol{w}^*_{(k)}, b^*_{0,(k)}, b^*_{1,(k)} = \underset{\boldsymbol{w}, b_0, b_1}{\operatorname{argmin}} \ -\ell(\boldsymbol{w}, b_0, b_1) + \sum_{f \in \mathcal{F}_k} c_f w_f; \ s.t. \ w_f \geq 0, \ f \in \mathcal{F}_k. \quad (7)$$

This can be regarded as the *rule evaluation stage*, where all rules in the current set will be reweighed. Note that an *optimal* solution to the *restricted* master problem above can be extended to a *feasible* solution to the original problem by setting the weights of all missing $f \in \bar{\mathcal{F}} \setminus \mathcal{F}_k$ to zero. The *optimality* of this extended solution can be verified by leveraging the convexity property of the original problem (6). Let the Lagrangian of the original problem be $L(\boldsymbol{w}, b_0, b_1, \boldsymbol{\nu}) = -\ell(\boldsymbol{w}, b_0, b_1) + \sum_{f \in \bar{\mathcal{F}}} c_f w_f - \sum_{f \in \bar{\mathcal{F}}} \nu_f w_f$, where $\nu_f \geq 0$ is the Lagrange multiplier associated with the non-negativity constraints of w_f. Being a convex problem, strong duality holds under mild conditions. Suppose $\boldsymbol{w}^*, b_0^*, b_1^*$ is primal optimal, and $\boldsymbol{\nu}^*$ is dual optimal, then $-\ell(\boldsymbol{w}^*, b_0^*, b_1^*) = \inf_{\boldsymbol{w}, b_0, b_1} L(\boldsymbol{w}, b_0, b_1, \boldsymbol{\nu}^*)$. This implies the *complementary slackness*, which will lead to the following condition.

Evaluate current dual price. Obtain the dual price for each constraint in the *original problem (6)*:

$$\nu_{f,(k)} = -\left.\frac{\partial \ell(\boldsymbol{w}, b_0, b_1)}{\partial w_f}\right|_{\boldsymbol{w}^*_{(k)}, b^*_{0,(k)}, b^*_{1,(k)}} + c_f, \quad (8)$$

where subscript (k) means the price depends on the current optimal solution of the RMP, and subscript f refers to the constraint $w_f \geq 0$. The optimality condition says: if the extended solution of $\boldsymbol{w}^*_{(k)}$ is optimal to the original problem, it must satisfy the following condition:

$$w^*_{f,(k)} > 0 \ \Rightarrow \ \nu_{f,(k)} = 0; \qquad w^*_{f,(k)} = 0 \ \Rightarrow \ \nu_{f,(k)} \geq 0 \qquad \forall f \in \bar{\mathcal{F}}, \quad (9)$$

where $\nu_{f,(k)}$ is computed via (8). For the missing rules, we have $w_{f,(k)} = 0$ automatically, and if they are the optimal solution to the original problem, the computed $\nu_{f,(k)} \geq 0$ must hold. A more detailed description of complementary slackness and optimality condition is given in Appendix D.

So, the rule adding criterion is to search over the missing rules and find the rule that most *violates* the condition, i.e., leads to the most negative ν_f by (8). Adding it to $\mathcal{F}_{k+1}$ will most reduce the objective value. Assuming the computed minimal $\nu_f \geq 0$, it is guaranteed that the current solution to the restricted problem is also optimal to the original one. A subproblem is constructed and optimized to identify a new rule, as will be discussed next.

3.3 Propose a New Temporal Logic Rule

Construct a subproblem from the current dual price. A subproblem is formulated to propose a new (or missing) temporal logic rule which can potentially improve the optimal value of the RMP, i.e. having a negative increased gain. The increased gain of a missing rule is defined as the possible change in the objective per unit when it is included in the rule set, and it is computed by taking

the partial derivative of the objective function w.r.t the rule weight. Note the subproblem itself is a minimization, to find the most negative increased gain.

Given the solution $\boldsymbol{w}^*_{(k)}, b^*_{0,(k)}, b^*_{1,(k)}$ for the restricted master problem (7), a subproblem is constructed by taking the partial derivative w.r.t the weights. The log likelihood can be regarded as a function of the dual conditional intensities: $\ell(\lambda^*, \mu^*)$. By the chain rule, the subproblem optimizes:

$$\min_{\phi_f \in \Phi} \nu(\phi_f | \boldsymbol{w}^*_{(k)}, b^*_{0,(k)}, b^*_{1,(k)}) = -\left(\frac{\partial \ell(\lambda^*, \mu^*)}{\partial \lambda^*} \frac{\partial \lambda^*}{\partial w_f} + \frac{\partial \ell(\lambda^*, \mu^*)}{\partial \mu^*} \frac{\partial \mu^*}{\partial w_f} \right) \Bigg|_{\boldsymbol{w}^*_{(k)}, b^*_{0,(k)}, b^*_{1,(k)}} + c_f, \quad (10)$$

where $\phi_f \in \Phi$ is the rule-based feature, λ^* and μ^* are functions of ϕ_f, and the conditional intensities are time and history-dependent (see Eq. (4)). In our formulation, this subproblem objective happens to be the current dual price. The objective can be hard to express in a closed-form but will be easily evaluated numerically. Minimizing over the feature space Φ is essentially searching through a combinatorial space of the temporal predicates and their temporal relations to determine a rule f, so that its yielding feature will minimize the objective function.

Solve subproblem to identify a new rule.

As discussed above, if the optimal value of the subproblem is negative, we have identified a feature (i.e., rule) to enter the rule set to construct $\mathcal{F}_{k+1}$; if the optimal value of the above problem is non-negative, we have proven that the current solution to the RMP is also optimal to the original problem. Note that the subproblem would not generate the same feature or rule more than once, since the optimal Lagrangian multipliers in the RMP are non-negative. However, explicitly solving subproblem (10) requires enumerating all possible conjunctions of the input predicates, their signs, and all possible pairwise temporal relations. The construction of feature ϕ_f requires evaluating all valid combinations of predicate events from history.

Speedup subproblem. To make the subproblem tractable, we adopt a couple of principles. 1) Theoretically, the *optimality of the subproblem can be sacrificed*. In fact, any solution to (10) that has *a negative objective* will generate a rule with negative increased gain and can be added to the rule set. This provides a performance guarantee for speedup. 2) Practically, *prior knowledge in rule structures can be leveraged*. Our prior knowledge has 3 parts: *sparsity*, *heredity*, and *expert's preference* in rule templates. *sparsity* means the uncovered rule set is small and the rule lengths are generally short. In our algorithm, the maximal rule length is pre-specified as H. It means we do not need to consider rules with lengths exceeding H. By *heredity*, we mean if the conjunction of the input predicates is important, at least one of the involved input predicates will show some significance. Heredity justifies that we can gradually grow rules from the existing short segments in the current set.

Search algorithms. With the prior knowledge above, we can design search algorithms in a way like depth-first search (DFS) or breadth-first search (BFS). In our setting, they are named as rule-extension-first-search (REFS) and rule-addition-first-search (RAFS), respectively. REFS will always extend the existing rules until the length reaches H, while RAFS will first add all possible rules with length 1, and then consider the rules with length 2 and so on. Here we describe details of REFS, and leave RAFS for Appendix F. In Fig. 4, a REFS is used to construct a new temporal logic rule. Specifically, it starts by constructing a rule with length one (i.e., one body predicate). This is achieved by scanning the predicate set $\mathcal{X}$ and temporal relation set $\mathcal{R}$, then enumerating the signs and possible temporal relations with the head predicate. The subproblem objective is used to score these segments and select the most negative one. The process continues to extend current segments in the same way, until the rule length reaches H, or the minimal subproblem objective becomes positive. In this way, the complexity of our subproblem is $\mathcal{O}(d)$, where d is the number of predicates. Once new rules are added to the rule set, the model will be re-fitted and the model parameters will be re-weighted according to Eq. (7). Longer rules might outweigh their short segments after the rule evaluation stage and thus the length of an important rule can be determined.

In addition, we can use *expert's preference* in rule templates to solve the subproblems. Human experts provide preferred rule templates (rather than a rule set), which reduce the rule searching space, since it masks out many irrelevant predicates and their combinations. For example, doctors are interested in how a treatment affects the evolution of patient health status. To reflect the doctor's conjectures, the rule template can be defined as "symptoms $\leftarrow$ drugs $\wedge$ symptoms".

3.4 Discussion on Scalability of the Algorithm

The previous discussion provides a theoretical framework to find an (near-) optimal temporal logic rule set based on the point process model. For large problems with thousands of event sequences and hundreds of predicates, this type of progressive algorithm tends to be hard to terminate, and obtaining

the optimality condition is almost impossible; this is the so-called tailing-off effect (Lübbecke & Desrosiers, 2005; Savelsbergh, 2002). Therefore, we need other techniques to make the TELLER scale with the data and the predicates. For example, we may condition an early termination to the algorithm if a fixed time limit is exceeded. In practice, setting the time limit is proven to work well and the most important rules are likely to be discovered. For large problems, we may also choose to take a random subset of sequences and a random subset of predicates for evaluation. When solving for the RMP, we can use a stochastic gradient descent-type of optimization to approximate the gradient using a batch of sequences. By doing so, we enable the RMP to scale with the data (sequences). Furthermore, we can warm-start the model parameters for consecutive RMPs, i.e. we initialize $\boldsymbol{w}_{(k+1)}, b_{0,(k+1)}, b_{1,(k+1)}$ using the optimal solution $\boldsymbol{w}^*_{(k)}, b^*_{0,(k)}, b^*_{1,(k)}$. When solving the subproblems, besides the efficient search scheme, we may also choose to take a subset of the predicates in searching for further speedup.

When the domain knowledge is not accessible, a rule of thumb to initialize $\mathcal{F}_0$ is to start with an "empty" set, which refers to there being only a base term in the intensity function, i.e., $\lambda^*(t) = \exp(b_0)$ and the initial estimate can be learned by MLE (i.e., restricted master problem).

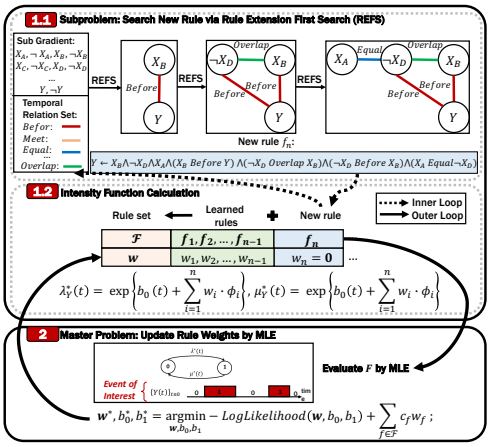

Figure 4: The overall flow of TELLER. It alternates between a rule evaluation stage (outer loop in the figure) and a rule proposal stage (inner loop). The set of logic rules and their importance weights are jointly learned.

4 Experiments

We evaluate the rule discovery ability and prediction accuracy of TELLER on synthetic and real event data. For synthetic data, with ground-truth temporal logic rules, the accuracy can be evaluated by comparing the uncovered results to the ground truth. For real data, we consider a healthcare treatment understanding problem and a crime pattern learning problem (crime results are in Appendix H & I).

4.1 Baselines

We considered the following SOTA baselines: 1) Recurrent Marked Temporal Point Processes (RMTPP) (Du et al., 2016b), the first neural point process (NPP) model, where the intensity function is modeled by a Recurrent Neural Network (RNN); 2) Neural Hawkes Process (NHP) (Mei & Eisner, 2016), an improved variant of RMTPP by constructing a continuous-time LSTM; 3) Transformer Hawkes Process (THP) (Zuo et al., 2020), an NPP model with a self-attention mechanism; 4) Tree-Regularized GRU (TR-GRU) (Wu et al., 2018a), a deep time-series model with a designed tree regularizer to add model interpretability; 5) Recurrent Point Process Network (RPPN) (Xiao et al., 2019), where the intensity is modeled by two interleaved RNNs with attention mechanism; 6) CAUSE (Zhang et al., 2020b), an NPP model with Granger causality statistic learned. We also considered other widely used parametric/nonparametric TPPs: 1) Hawkes Process with an exponential kernel (HExp) (Lewis & Mohler, 2011); 8) Inhomogeneous Poisson Process (IPP), where the intensity is a sum of k weighted Gaussian kernels: $\hat{\lambda}(t) = \sum_{i=1}^{k} \alpha_i \left(2\pi\sigma_i^2\right)^{-1/2} \exp\left(-\left(t-c_i\right)^2/\sigma_i^2\right)$, where c_i and σ_i are fixed center and standard deviation, and α_i is the weight for kernel i.

Table 1: Model properties and interpretability comparison of all baselines and TELLER

Model	RMTPP	NHP	THP	TR-GRU	RPPN	CAUSE	HExp	IPP	**TELLER**
Interpretable				✓		✓	✓	✓	✓
Flexible	✓	✓	✓	✓	✓	✓			✓
Parsimonious							✓	✓	✓
Pairwise interaction						✓	✓		✓
Higher-order interaction									✓

In Tab. 1, we compared TELLER with these baselines by the model properties: 1) Interpretable: model parameters and prediction are understandable to humans; 2) Flexible: model is expressive to capture sophisticated non-linear dependencies; 3) Parsimonious: model parameters are of small or medium size; 4) Pairwise interaction: can identify the mutual influence of the events; 5) Higher-order interaction: can identify the composite effect of a subset of events to another event.

4.2 Synthetic Data

We first verify TELLER's rule discovery ability on synthetic datasets with ground truth. The synthetic events are generated from TLPPs with a known set of rules and weights. We prepared 12 synthetic

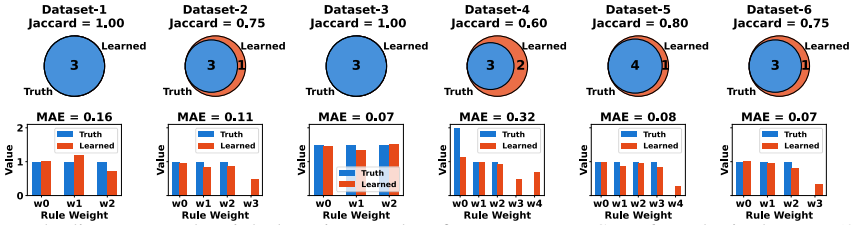

Figure 5: Rule discovery and weight learning results of TELLER-REFS on 6 synthetic datasets (2.4K seqs).

datasets by specifying various rule lengths, weights, and temporal relations. See Appendix G for details. Both the REFS and RAFS subproblem search schemes are tested on all 12 datasets. Consistent results show an accurate performance of our algorithm in terms of both the rule discovery and parameter learning. We list all the discovered rules based on dataset-1 using 2400 and 600 sequences respectively in Tab. 2, where the correct rules are marked in blue. For dataset-1, the body predicate set is {A, B, C, D}, and the head predicate is E. Given 600 sequences, TELLER successfully discovered 2 out of 3 truth rules along with several noise rules; whereas, given 2400 sequences, TELLER accurately uncovered all 3 ground truth rules without including any noise rules. The results indicate that TELLER is capable of mining hidden rules from noisy event data. Given more samples, the learning performance will improve as a result.

Table 2: Learned Rules on Synthetic Dataset-1.

True Rules (# 3) v.s. Learned Rules (REFS)
2400 Sequences:
E ← A ∧ (A Before E)
E ← B ∧ C ∧ (B Before E) ∧ (C Before E)
E ← C ∧ D ∧ (C Before D) ∧ (D Equal E)
600 Sequences:
E ← A ∧ (A Before E)
E ← A ∧ D ∧ (A Equal D) ∧ (A Before E)
E ← A ∧ ¬B ∧ (A Before E) ∧ (¬B Before A)
E ← B ∧ C ∧ (B Before E) ∧ (C Before E)
E ← B ∧ C ∧ (B Before E) ∧ (C Before B)
E ← A ∧ C ∧ (A Before E) ∧ (C Before E)
E ← C ∧ A ∧ (A Equal C) ∧ (A Before E)
E ← A ∧ ¬C ∧ (A Before ¬C) ∧ (¬C Before E)

More datasests' results are reported in Fig. 5, where 6 synthetic datasets with 2400 sequences are used for evaluation. Each plot in the top row uses a Venn diagram to show the true rule set and the learned rule set, from which the Jaccard similarity score (area of the intersection divided by the area of their union) is calculated. TELLER discovered almost all the true rules. Each plot in the bottom compares the true rule weights with the learned rule weights, with the Mean Absolute Error (MAE) reported. Almost all truth rule weights are accurately learned and the included noise rule weights are relatively small. In dataset-4, we crafted a long and complex rule with 3 body predicates to challenge TELLER. It is still able to correctly discover this rule, while the weight is slightly underestimated. The existence of long logic rules will require more data and stronger signals (i.e., higher rule weights) to have an accurate recovery. More information about the experiment settings can be found in Appendix G.

We also compared TELLER with the brute-force method, where we enumerated all possible temporal logic rules and built a full model to learn the rule weights. We showed the training process for dataset-1 in Fig. 6, where the evolving likelihood versus the run time is displayed. For TELLER, we also marked the time point when the true rule is discovered (gray dashed vertical lines). The key observations are: 1) For TELLER, the likelihood increases very fast and we can see evident jumps whenever a true rule is discovered. As a comparison, the likelihood for the brute-force method increases slowly. 2) All the true rules can be discovered efficiently by TELLER and this verifies that we can set a time limit for early stopping. We can stop TELLER when the objective function seems to start centering around some level. This experiment verifies that our method is computationally efficient compared to the brute-force methods.

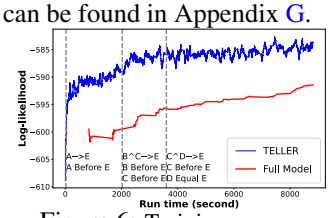

Figure 6: Training process.

4.3 Real Data: Treatment on MIMIC-III

MIMIC-III contains electronic health records of patients admitted to the intensive care unit (ICU) (Johnson et al., 2016). We focused on patients diagnosed with sepsis (Saria, 2018; Raghu et al., 2017; Peng et al., 2018), as one of the major causes of mortality in ICU. Evidence suggests that the treatment strategy remains uncertain – it is unclear how to use intravenous fluids and vasopressors to support the circulatory system. There also exists clinical controversy about when and how to use these two groups of drugs to reduce the side effect. TELLER is aimed to learn the explanatory temporal logic rules regarding these two groups of drugs as well as their importance weights.

Predicates and dataset statistics. We define 62 predicates, including two groups of drugs (i.e., intravenous fluids and vasopressors) and lab measurements. The variables involved are similar to (Saria, 2018). Among all these predicates, we are interested in reasoning about two predicates. So we define them as head predicates: 1) `LowUrine` and 2) `Survival`. We treat real-time urine as a

head predicate since low urine is the direct indicator of bad circulatory systems and is an important signal for septic shock. A complete table of predicates can be found in Appendix H. In our experiment, lab measurement variables are converted to binary values (according to the normal range used in medicine) with the transition times recorded. For drug predicates, they are recorded as 1 when they were applied to the patient. We extracted 4298 patient sequences, and randomly chose 80% of them for training and the remaining for testing. The average time horizon is 392.69 hours and the average #events per sequence is 79.03. Detailed setup and hyperparameters can be found in Appendix H.2.

Evaluation metrics and results. We use different evaluation metrics for the two head predicates: 1) `LowUrine` is evaluated by predicting the time of transitions from state 0 to state 1 (this is related to the appearance of septic shock in medicine), for which the performance is measured by Mean Absolute Error (MAE). 2) `Survival` is evaluated by predicting whether the patient is survived at the end of treatment, for which the performance is measured by AUC. The performances of our model and all baselines are given in Tab. 3.

Table 3: MIMIC-III: Event prediction results.

Method	LowUrine (MAE)	Survival (AUC)
RMTPP	1.983	0.796
NHP	1.684	0.802
THP	1.545	0.824
TR-GRU	2.666	0.811
RPPN	1.640	0.767
CAUSE	1.712	0.740
HExp	2.578	0.882
IPP	2.472	0.784
TELLER	**1.266**	**0.930**

Discovered temporal logic rules. Tab. 4 shows the uncovered explanatory temporal logic rules and their weights learned by TELLER. Here, we use `LowUrine` as the head predicate and `NormalUrine` as its negation. For each discovered rule, the sign of the head predicate will be automatically determined by the algorithm. Human experts (i.e., doctors in ICU) confirmed that these discovered logic rules are consistent with the pathogenesis of sepsis, and have captured the most important factors in affecting real-time urine output and survival. Experts are also interested in interpreting the clinical meaning of the discovered rules. They found that Rule 1 to Rule 4 capture the major lab measurements that usually emerge together with extremely low urine. Rule 5 seemed hard to interpret in the beginning by experts, since they thought it was slightly counter-intuitive. They later concluded that, in some extreme cases, even after using Crystalloid (one type of intravenous fluids) as the treatment, the urine can remain low, and thus Rule 5 is still likely to be observed. Rules 6-9 shed light on drug selection. For example, reflected in Rule 7, using Crystalloid and Phenylephrine (a type of vasopressors) together will play an evident role in triggering the urine to a normal level. In Rules 6, 8, and 9, using Crystalloid yields a weight of 4.14, using Colloid (a type of intravenous fluids) yields a weight of 1.97, and using the combinations of the two yields a weight of 1.83, which can be regarded as an incremental contribution in addition to using them separately.

Expert-Model interaction. Experts are involved in TELLER's learning loop: 1) Experts can provide a predicate set, their interested head predicates, an initial rule set, and desired rule templates to the model; 2) Run TELLER to obtain preliminary results for experts to check. Experts can delete unreasonable or noise rules and give the refined rule set to the model before continuing learning. Experts provided advice for predicates definition and the desired rule templates. We ran TELLER from an empty set and generated 12 rules. Experts verified the rules' correctness and deleted the following ones: `LowUrine` $\leftarrow$ `HighSpO2`; `LowUrine` $\leftarrow$ `NormalSodium`; `LowUrine` $\leftarrow$ `NormalSodium` $\wedge$ `Water`. We were informed that the first rule is contradictory with Pathophysiology common sense, while the other two rules are correct yet irrelevant to the treatment strategy. We deleted these 3 rules and re-implemented TELLER to refine the rule set and weights with final output in Tab. 4.

Table 4: Learned Rules with `LowUrine` as the head predicate.

Weight	Rule
1.07	**Rule 1:** LowUrine $\leftarrow$ LowSysBP $\wedge$ (LowSysBP Before LowUrine)
0.89	**Rule 2:** LowUrine $\leftarrow$ LowSodim $\wedge$ (LowSodim Before LowUrine)
1.16	**Rule 3:** LowUrine $\leftarrow$ HighCreatinine $\wedge$ Equal(HighCreatinine, LowUrine)
1.98	**Rule 4:** LowUrine $\leftarrow$ HighBUN $\wedge$ (HighBUN Before LowUrine)
1.10	**Rule 5:** LowUrine$\leftarrow$ HighBUN $\wedge$ Crystalloid $\wedge$ (HighBUN Before LowUrine) $\wedge$ (Crystalloid Before LowUrine)
4.14	**Rule 6:** NormalUrine $\leftarrow$ LowUrine $\wedge$ Crystalloid $\wedge$ (LowUrine Equal NormalUrine) $\wedge$ (Crystalloid Equal NormalUrine)
3.13	**Rule 7:** NormalUrine $\leftarrow$ LowUrine $\wedge$ Phenylephrine $\wedge$ (LowUrine Equal NormalUrine) $\wedge$ (Phenylephrine Equal NormalUrine)
1.83	**Rule 8:** NormalUrine $\leftarrow$ LowUrine $\wedge$ Crystalloid $\wedge$ Colloid $\wedge$ (Colloid Equal NormalUrine) $\wedge$ (Crystalloid Equal NormalUrine) $\wedge$ (LowUrine Equal NormalUrine)
1.97	**Rule 9:** NormalUrine $\leftarrow$ LowUrine $\wedge$ Colloid $\wedge$ (LowUrine Equal NormalUrine) $\wedge$ (Colloid Equal NormalUrine)

5 Conclusion

In this paper, we have proposed a new algorithm TELLER to learn interpretable temporal logic rules to explain point processes, with promising results on both synthetic and real-world data. To our best knowledge, this is the first work that automates the discovery of the temporal logic rules based on point process models. Our method adds transparency and interpretability to event models.

REFERENCES

James F Allen. Maintaining knowledge about temporal intervals. In *Readings in qualitative reasoning about physical systems*, pp. 361–372. Elsevier, 1990.

Cynthia Barnhart, Ellis L Johnson, George L Nemhauser, Martin WP Savelsbergh, and Pamela H Vance. Branch-and-price: Column generation for solving huge integer programs. *Operations research*, 46(3):316–329, 1998.

William Brendel, Alan Fern, and Sinisa Todorovic. Probabilistic event logic for interval-based event recognition. In *CVPR 2011*, pp. 3329–3336. IEEE, 2011.

William W Cohen. Tensorlog: A differentiable deductive database. *arXiv preprint arXiv:1605.06523*, 2016.

Sanjeeb Dash, Oktay Gunluk, and Dennis Wei. Boolean decision rules via column generation. *Advances in Neural Information Processing Systems*, 31:4655–4665, 2018.

N. Du, H. Dai, R. Trivedi, U. Upadhyay, M. Gomez-Rodriguez, and L. Song. Recurrent marked temporal point processes: Embedding event history to vector. In *KDD*, 2016a.

Nan Du, Hanjun Dai, Rakshit Trivedi, Utkarsh Upadhyay, Manuel Gomez-Rodriguez, and Le Song. Recurrent marked temporal point processes: Embedding event history to vector. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 1555–1564, 2016b.

Richard Evans and Edward Grefenstette. Learning explanatory rules from noisy data. *Journal of Artificial Intelligence Research*, 61:1–64, 2018.

Ludwig Fahrmeir and Gerhard Tutz. *Multivariate statistical modelling based on generalized linear models*. Springer Science & Business Media, 2013.

Bernd Finkbeiner, Christopher Hahn, Markus N Rabe, and Frederik Schmitt. Teaching temporal logics to neural networks. *arXiv preprint arXiv:2003.04218*, 2020.

Valentin Goranko and Antje Rumberg. Temporal Logic. In Edward N. Zalta (ed.), *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, Fall 2021 edition, 2021.

A. Hawkes. Point spectra of some mutually exciting point processes. *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 438–443, 1971a.

Alan G Hawkes. Spectra of some self-exciting and mutually exciting point processes. *Biometrika*, 58 (1):83–90, 1971b.

V. Isham and M. Westcott. A self-correcting point process. *Stochastic Processes and Their Applications*, 8(3):335–347, 1979.

Alistair EW Johnson, Tom J Pollard, Lu Shen, H Lehman Li-Wei, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G Mark. Mimic-iii, a freely accessible critical care database. *Scientific data*, 3(1):1–9, 2016.

E. Lewis and E. Mohler. A nonparametric em algorithm for multiscale hawkes processes. *Journal of Nonparametric Statistics*, 2011.

Shuang Li, Lu Wang, Ruizhi Zhang, Xiaofu Chang, Xuqin Liu, Yao Xie, Yuan Qi, and Le Song. Temporal logic point processes. In *International Conference on Machine Learning*, pp. 5990–6000. PMLR, 2020.

Xin Liu, Junchi Yan, Shuai Xiao, Xiangfeng Wang, Hongyuan Zha, and Stephen Chu. On predictive patent valuation: Forecasting patent citations and their types. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017.

Yu-Ying Liu, Shuang Li, Fuxin Li, Le Song, and James M Rehg. Efficient learning of continuous-time hidden markov models for disease progression. *Advances in neural information processing systems*, 28:3599, 2015.

Marco E Lübbecke and Jacques Desrosiers. Selected topics in column generation. *Operations research*, 53(6):1007–1023, 2005.

Hongyuan Mei and Jason Eisner. The neural hawkes process: A neurally self-modulating multivariate point process. *arXiv preprint arXiv:1612.09328*, 2016.

Boston Department of Innovation and Technology. Crime Incident Reports. https://data.boston.gov/dataset/crime-incident-reports-august-2015-to-date-source-new-system, 2015. [Online; accessed 03-May-2021].

Tivadar Papai, Henry Kautz, and Daniel Stefankovic. Slice normalized dynamic markov logic networks. *Advances in Neural Information Processing Systems*, 25:1907–1915, 2012.

Xuefeng Peng, Yi Ding, David Wihl, Omer Gottesman, Matthieu Komorowski, Li-wei H Lehman, Andrew Ross, Aldo Faisal, and Finale Doshi-Velez. Improving sepsis treatment strategies by combining deep and kernel-based reinforcement learning. In *AMIA Annual Symposium Proceedings*, volume 2018, pp. 887. American Medical Informatics Association, 2018.

Aniruddh Raghu, Matthieu Komorowski, Imran Ahmed, Leo Celi, Peter Szolovits, and Marzyeh Ghassemi. Deep reinforcement learning for sepsis treatment. *arXiv preprint arXiv:1711.09602*, 2017.

Matthew Richardson and Pedro Domingos. Markov logic networks. *Machine learning*, 62(1-2): 107–136, 2006.

Ali Sadeghian, Mohammadreza Armandpour, Patrick Ding, and Daisy Zhe Wang. Drum: End-to-end differentiable rule mining on knowledge graphs. *arXiv preprint arXiv:1911.00055*, 2019.

Suchi Saria. Individualized sepsis treatment using reinforcement learning. *Nature medicine*, 24(11): 1641–1642, 2018.

Martin WP Savelsbergh. Branch-and-price: Integer programming with column generation, bp. 2002.

Parag Singla and Pedro M Domingos. Lifted first-order belief propagation. In *AAAI*, volume 8, pp. 1094–1099, 2008.

Son D Tran and Larry S Davis. Event modeling and recognition using markov logic networks. In *European Conference on Computer Vision*, pp. 610–623. Springer, 2008.

Po-Wei Wang, Priya Donti, Bryan Wilder, and Zico Kolter. Satnet: Bridging deep learning and logical reasoning using a differentiable satisfiability solver. In *International Conference on Machine Learning*, pp. 6545–6554. PMLR, 2019a.

Po-Wei Wang, Daria Stepanova, Csaba Domokos, and J Zico Kolter. Differentiable learning of numerical rules in knowledge graphs. In *International Conference on Learning Representations*, 2019b.

Tong Wang, Cynthia Rudin, Finale Doshi-Velez, Yimin Liu, Erica Klampfl, and Perry MacNeille. A bayesian framework for learning rule sets for interpretable classification. *The Journal of Machine Learning Research*, 18(1):2357–2393, 2017.

Dennis Wei, Sanjeeb Dash, Tian Gao, and Oktay Gunluk. Generalized linear rule models. In *International Conference on Machine Learning*, pp. 6687–6696. PMLR, 2019.

Mike Wu, Michael Hughes, Sonali Parbhoo, Maurizio Zazzi, Volker Roth, and Finale Doshi-Velez. Beyond sparsity: Tree regularization of deep models for interpretability. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018a.

Weichang Wu, Junchi Yan, Xiaokang Yang, and Hongyuan Zha. Decoupled learning for factorial marked temporal point processes. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 2516–2525, 2018b.

Shuai Xiao, Junchi Yan, Mehrdad Farajtabar, Le Song, Xiaokang Yang, and Hongyuan Zha. Learning time series associated event sequences with recurrent point process networks. *IEEE transactions on neural networks and learning systems*, 30(10):3124–3136, 2019.

Junchi Yan, Shuai Xiao, Changsheng Li, Bo Jin, Xiangfeng Wang, Bin Ke, Xiaokang Yang, and Hongyuan Zha. Modeling contagious merger and acquisition via point processes with a profile regression prior. In *IJCAI*, pp. 2690–2696, 2016.

Junchi Yan, Hongteng Xu, and Liangda Li. Modeling and applications for temporal point processes. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 3227–3228, 2019.

Fan Yang, Zhilin Yang, and William W Cohen. Differentiable learning of logical rules for knowledge base reasoning. *arXiv preprint arXiv:1702.08367*, 2017.

Yuan Yang and Le Song. Learn to explain efficiently via neural logic inductive learning. In *International Conference on Learning Representations*, 2019.

Qiang Zhang, Aldo Lipani, Omer Kirnap, and Emine Yilmaz. Self-attentive hawkes process. In *International Conference on Machine Learning*, pp. 11183–11193. PMLR, 2020a.

Qiang Zhang, Aldo Lipani, and Emine Yilmaz. Learning neural point processes with latent graphs. In *Proceedings of the Web Conference 2021*, pp. 1495–1505, 2021.

Wei Zhang, Thomas Panum, Somesh Jha, Prasad Chalasani, and David Page. Cause: Learning granger causality from event sequences using attribution methods. In *International Conference on Machine Learning*, pp. 11235–11245. PMLR, 2020b.

Simiao Zuo, Haoming Jiang, Zichong Li, Tuo Zhao, and Hongyuan Zha. Transformer hawkes process. In *International Conference on Machine Learning*, pp. 11692–11702. PMLR, 2020.