Reinforcement Temporal Logic Rule Learning to Explain the Generating Processes of Events

Chao Yang [1] Lu Wang [2] Zhaner Mou [1] Shuang Li [1]

Abstract

Understanding the generating process of events with irregular timestamps has long been an interesting problem. Recent advances in neural-based event models have exhibited superior ability in event prediction. However, the lack of interpretability of these black-box models hinders their applications in high-stakes systems like healthcare. In this paper, we propose an explainable and flexible event model. We first model the event data by multivariate temporal point processes, where the intensity functions are informed by a set of temporal logic rules. We then propose a learning algorithm that automates the rule discovery process from data to explain the event data. Specifically, we formulate the rule discovery problem as a reinforcement learning problem, where a recurrent neural network type of policy is trained to search for the best-explanatory temporal logic rules. A reward function is designed so that the learned policy can generate the temporal logic rules that best explain the event data. We evaluate our methods on both synthetic and real-world datasets, obtaining promising results.

1. Introduction

Nature or complex dynamic systems produce large volumes of discrete event data with irregular timestamps, such as crimes, earthquakes, power failures, urban fires, and contagious diseases. It is essential to understand the dynamics of these event data so that one can predict the “time-to-event", perform abnormal event detection, or execute smart intervention. Among many event models, temporal point process (TPP) can model event data in continuous time. Instead of discretizing the time horizons and converting the event data into time-series event counts, TPP models treat the inter-event times as random variables and directly model their probability distributions. TPP model is elegant and can accommodate the irregularly spaced discrete events. The learned model can readily predict the *time-to-event* as well as the *event types*.

Mathematically, TPP models can be characterized by the intensity function, which quantifies how many events will occur within a short time interval. The modeling framework boils down to the design of the intensity function, which can incorporate features relative to seasonality, location, or temporally related covariates to add the flexibility and interpretability of the model (Mohler et al., 2011). Recent development in deep learning has benefited TPP models. For example, (Du et al., 2016) proposed the neural point process model, named RMTPP, where the intensity function is modeled by a Recurrent Neural Network. (Mei & Eisner, 2017) improved RMTPP by constructing a continuous-time RNN. (Zuo et al., 2020) and (Zhang et al., 2020a) recently leveraged the self-attention mechanism to capture the long-term dependencies of events. These neural-based TPP models have achieved remarkable performance in event time and event type prediction.

Although flexible, these neural TPP models are black-box and are hard to interpret. To add transparency, recently (Zhang et al., 2020b) introduced Granger causality as a latent graph to explain point processes, and the structures are jointly learned via gradient descent. However, Granger causality is still limited to the triggering patterns of events. To preserve the model interpretability and enable the model to capture more sophisticated and nonlinear dependency patterns, (Li et al., 2020) proposed an explainable Temporal Logic Point Process (TLPP), where the construction of the intensity function is informed by a set of pre-specified temporal logic rules. TLPP model enables one to perform symbolic reasoning based on TPP. A follow-up work (Li et al., 2021) further designed a branch and price algorithm that can learn the set of logic rules and the rule weights, which further enhances the flexibility of the model.

TLPP is a sound model that enables one to use logical reasoning to induce the occurrence of events. For example, we can infer whether a person gets covid by the following

[1] School of Data Science, The Chinese University of Hong Kong (Shenzhen), China [2] Microsoft Research. Correspondence to: Shuang Li <lishuang@cuhk.edu.cn>.

Proceedings of the 39^{th} International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022. Copyright 2022 by the author(s).

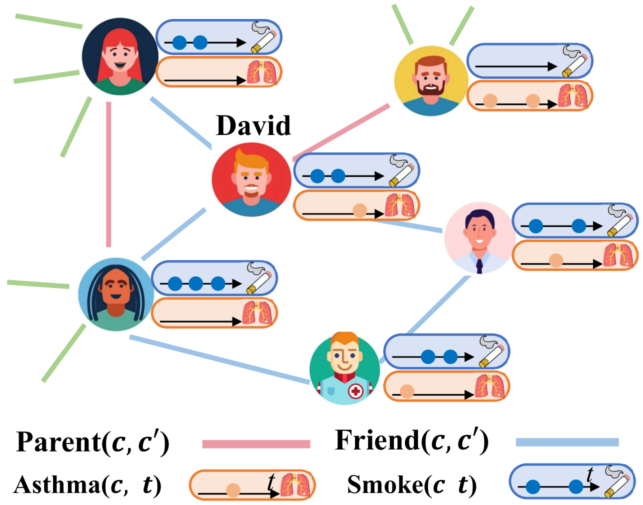


Figure 1. The triggering mechanism of smoking and asthma is complex and intertwined. The mechanism can be expressed by the following logic rules: $f_1 : \forall c, Asthma(c,t) \leftarrow Smokes(c,t') \wedge AsthmaGene(c,t'') \wedge (Smokes(c,t')\ After\ AsthmaGene(c,t''))$, $f_2 : \forall c, \forall c', Smokes(c',t) \leftarrow Friend(c,c') \wedge Smokes(c,t')$, $f_3 : \forall c, AsthmaGene(c',t) \leftarrow Parent(c,c') \wedge Asthma(c,t')$, where t, t' and t'' indicate different time points. All body conditions must happen before the head predicate gets fired.

temporal logic rule – “if a person once got exposed to the coronavirus and afterward exhibited related symptoms, (e.g., headache, cough), then he/she is likely to get covid". Temporal logic rules are logical connectives of predicates with temporal ordering constraints. Predicates are boolean logical variables, describing the property or relation of entities. For example, the action that the person "once got exposed to the coronavirus" and the observation that the person "exhibited related symptoms" can be defined as the property predicates of the entity (i.e., a person). There can exist extra temporal ordering constraints such as "afterward" in the rule defining the temporal ordering condition that the evidence must satisfy. The logic rule describes how could we perform the event induction from the evidence to the target event.

In this paper, we extend the original TLPP model to accommodate scenarios where the entity-entity relation is also known. For real-world events, such as disease outbreaks or social network events, data are usually produced by multiple entities having various relation types. Therefore, the events generated by different entities should be intertwined in a way that is subject to the underlying networks. For example, as illustrated in Fig. 1 the triggering mechanism of smoking and asthma is complex and intertwined. Asthma is a highly heritable disease. David was born to have the high risk of developing asthma because his father got asthma. David may also become a smoker if the people around them, like his friends, start smoking. Then David is more likely to develop into asthma, with a joint effect from that he starts smoking recently stimulated by his friends, and by the asthma gene inherited from his father. In this paper, we propose a unified TLPP modeling framework to accommodate the sophisticated triggering mechanism coming from different entity-entity networks.

What’s more challenging is to automatically uncover the important temporal logic rules and their weights from data. Each rule can be of various lengths, and can be defined by the logical connectives of property related predicates, entity-entity relations, and temporal ordering relations. Our search space is both discrete in model structure (rule structures) and continuous in model parameters (rule weights). The search space is usually huge, which grows exponentially with the number of predicates. This search problem is known to be NP-hard. In this paper, to speed up the learning process and tackle the scalability, we propose an efficient reinforcement learning algorithm.

Specifically, we propose an efficient neural policy-guided rule search algorithm to learn these explanatory temporal logic rules from large-scale noisy events, which conquers the big problem into tractable subproblems. We use a recurrent neural network to emit a distribution over tractable logic predicates and temporal relation expressions and employ a policy gradient to train the network to generate better explanatory logic rules. The method has the following advantages: 1) We make the temporal logic rule search problem differentiable, and all the policy parameters can be learned end-to-end. 2) The neural policy is expressive and has the ability to memorize and refine its predicate selection distributions. We hope in the formulated problem, the neural policy can learn to improve and put more priors to the more important predicates after iterations, which will excel the exhaustive search in speed in the long run. 3) Domain knowledge can be easily incorporated as constraints by designing a dynamic mask applied to the predicate selection distributions at each step.

Contributions Our main contributions have the following aspects: *i)* We propose a unified TLPP modeling framework to accommodate the sophisticated entity-entity relation networks. The new model can not only leverage the entity’s historical events to deduce new events but can also aggregate the entity’s various neighbors’ historical events. *ii)* We propose an efficient and differentiable policy-guided rule search algorithm to learn these explanatory temporal logic rules from noisy events over various entity-entity networks, which conquers the big problem into tractable subproblems. This neural policy can gradually update its predicted selection priors given the reward and will excel in speed compared to exhaustive search. *iii)* Finally, we evaluate our method on both synthetic and real-world datasets.

2. Background

2.1. Multivariate Temporal Point Processes

Let's start with the basic knowledge of TPP. TPP provides an elegant tool to capture the dynamics of the event sequences in continuous time. Given an event sequence

$$\mathcal{H}_t = \{t_1, t_2, \dots, t_n | t_n < t\}$$

up to t, which yields a counting process $\{N(t), t \geq 0\}$, the dynamics of the TPP can be characterized by conditional intensity function, denoted by $\lambda(t|\mathcal{H}_t)$. 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 $[t, t+dt]$. Given the definition of the intensity function, by some simple proof (Rasmussen, 2018), one can express the joint likelihood of the events $\mathcal{H}_t$ as

$$\begin{aligned} & p(\{t_1, t_2, \dots, t_n | t_n < t\}) \\ = & \prod_{t_i \in \mathcal{H}_t} \lambda(t_i|\mathcal{H}_{t_i}) \cdot \exp\left(-\int_0^t \lambda(\tau|\mathcal{H}_\tau)d\tau\right). \end{aligned} \tag{1}$$

Now, consider a U-dimensional multivariate TPP, $\{N_u(t), t \geq 0\}_{u=1,\dots,U}$, with corresponding intensity function $\{\lambda_u(t \mid \mathcal{H}_t)\}_{u=1,\dots,U}$ and $\mathcal{H}_t$ is the history of the entire multivariate events. One of the most famous multivariate TPP is the multivariate Hawkes process, which captures the mutual- and self-exciting triggering patterns of the events. The intensity function of multivariate Hawkes process is modeled as

$$\lambda_u(t \mid \mathcal{H}_t) = \mu_u + \sum_{u=1}^{U} \int_0^t g_{uu'}(t-s)dN_{u'}(s) \tag{2}$$

where $g(t)$ is called impact function. Recent neural based event models start modeling $\lambda_u(t \mid \mathcal{H}_t)$ as a deep learning model, such as RNN, which greatly increases the model flexibility but the learned model is hard to interpret. In this paper, we will focus on TLPP, which is a flexible and explainable model.

2.2. Temporal Logic Rules

Predicate Define a set of entities $\mathcal{C} = \{c_1, c_2, ..., c_n\}$. The *predicate* is defined as the *property* or *relation* of entities, which is a logic function that is defined over the set of entities, i.e.,

$$x(\cdot) : \mathcal{C} \times \mathcal{C} \cdots \times \mathcal{C} \mapsto \{0, 1\}$$

For example, $Smokes(c)$ is the property predicate and $Friend(c, c')$ is the relation predicate.

Logic Rule A *first-order logic rule* is a logical connectives of predicates, such as:

$$\begin{aligned} f_1 &: \forall c, Asthma(c) \leftarrow Smokes(c) \wedge AsthmaGene(c) \\ f_2 &: \forall c, \forall c', Smokes(c') \leftarrow Friend(c, c') \wedge Smokes(c) \end{aligned}$$

Temporal Logic Rule We first add a temporal dimension to the predicates. Then the temporal logic rule is a logic rule with temporal ordering constraints. For example,

$$\begin{aligned} f_1 : \forall c, Covid(c, t_3) \leftarrow & \; Symptoms\_Appear(c, t_2) \\ & \wedge Exposed\_to\_Virus(c, t_1) \\ & \wedge Before(t_1, t_2) \end{aligned}$$

For discrete events, we consider three types of temporal relations.

$$\begin{aligned} Before(t_1, t_2) &= \mathbb{1}\{t_1 - t_2 < 0\}, \\ After(t_1, t_2) &= \mathbb{1}\{t_1 - t_2 > 0\}, \\ Equal(t_1, t_2) &= \mathbb{1}\{t_1 = t_2\} \end{aligned}$$

We also treat the temporal relation as the temporal predicate, which is also a boolean logic variable.

3. Model: General Temporal Logic Point Processes

The *grounded* temporal predicate $x(c, t)$, such as $Asthma(c, t)$, will be recorded as a sequence of discrete events, $\{t_1, t_2, \dots, t_n | t_n < t\}$ up to t. The event sequence records the time points that the predicate gets fired and jumps to 1.

3.1. Logic-informed intensity function.

The main idea of TLPP (Li et al., 2020) is that the generating process of the grounded predicate events should be governed by a set of temporal logic rules. One can leverage the temporal logic rules as prior knowledge to deduce the occurrence of new events. From the modeling perspective, the intensity function can be modeled from the temporal logic rules.

How to inform the intensity design by temporal logic rules? In the original TLPP paper (Li et al., 2020), they consider the following temporal logic rule, for each $c \in \mathcal{C}$

$$f : Y(c, t_y) \leftarrow \underbrace{\bigwedge_{X_u \in \mathcal{X}_f} X_u(c, t_u)}_{\text{associated predicates}} \underbrace{\bigwedge_{X_u, X_{u'} \in \mathcal{X}_f} \mathcal{T}_{uu'}(t_u, t_{u'})}_{\substack{\text{temporal relation} \\ \text{of the associated predicates}}}, \tag{3}$$

$$\forall c \in \mathcal{C}$$

where $Y(c, t_y)$ is the head predicate needs to be grounded by the data of entity c at time point t_y, $\mathcal{X}_f$ is the set of predicates defined in rule f, and the temporal relation of predicate u and u', denoted as $\mathcal{T}_{uu'}$, can take any relation from the set $\{Before, After, Equal, None\}$. Note that it can take $None$, indicating there is no temporal relation constraint between predicate u and u'. t_y, t_u, and t'_u are the associated occurrence times of the predicates.

To further build a connection to the intensity function, they also assume ← has a causal direction. The body part of the rule indicates the evidence to be gathered from *history* to deduce the state of the head predicate. By this assumption, it is only valid to consider the historical body predicate events happened before t_y, i.e., $t_y \geq t_u, t'_u$. The high-level idea is once the body part of the rule is satisfied from history, then the head predicate gets fired and it tends to happen. The intensity function is modeled by taking inspiration from the Hawkes process that once the body condition of the rule becomes true, the intensity function of the head predicate will be boosted immediately. We can further model that this impact may decay slowly as time goes on by introducing various decaying impact functions like the Hawkes process.

The TLPP considers far more complicated triggering patterns than the Hawkes process. Hawkes process only considers pairwise triggering patterns, such as the self- or mutual-triggering pattern. TLPP, however, can model complicated conditional triggering patterns. For example, it can model that only given some conditions C, event A can trigger event Y. If the condition C doesn't hold, even if event A happens, event Y will not be triggered. Moreover, by adding a negation sign to the head predicate, the TLPP can model the self-correcting like patterns. By this comparison, we can see that TLPP is indeed very flexible and can model sophisticated and content-aware dependency patterns of events.

3.2. Consider various types of entity-entity relation

In this paper, we consider a more complicated setting where the entity-entity relation information is also known. We extend the TLPP to consider the following more general temporal logic rule involving entity-entity relations. For $\forall c, c' \in \mathcal{C}$

$$f : Y(c, t_y) \leftarrow \underbrace{R_d(c, c')}_{\substack{\text{entity-entity} \\ \text{relation}}} \underbrace{\bigwedge_{X_u^c \in \mathcal{X}_f^c} X_u^c(c, t_u)}_{\substack{\text{property predicates} \\ \text{about entity } c}} \underbrace{\bigwedge_{X_v^{c'} \in \mathcal{X}_f^{c'}} X_v^{c'}(c', t_v)}_{\substack{\text{property predicates} \\ \text{about entity } c'}} \tag{4}$$

$$\underbrace{\bigwedge_{X_u, X_{u'} \in \mathcal{X}_f^c \cup \mathcal{X}_f^{c'}} \mathcal{T}_{uu'}(t_u, t_{u'})}_{\substack{\text{temporal relation of all} \\ \text{the associated predicates}}}, \quad \forall c, c' \in \mathcal{C}$$

where

- ○ $Y(c, t_y)$ is the head predicate evaluated by the data of entity c at time point t_y.
- ○ $R_d(c, c') \in \{0, 1\}$ is the entity-entity relation between entities c and c', and d is the relation type. If $R_d(c, c') = 1$, we say entity c' is the neighbor of entity c, with a type d relation.
- ○ $\mathcal{X}_f^c$ is the set of property predicates regarding entity c associated with rule f. Each element $X_u^c(c, t_u)$ needs to be grounded by the data of entity c at time point t_u.
- ○ Similarly, $\mathcal{X}_f^c$ is the set of property predicates regarding neighbor entity c' associated with rule f. Each element $X_u^{c'}(c', t_v)$ needs to be grounded by the data of neighbor entity c' at time point t_v.
- ○ $\mathcal{T}_{uu'} \in \{Before, After, Equal, None\}$ is the temporal relation of predicates X_u and $X_{u'}$, where the associated two predicates can take any pair from the union of the predicate sets $\mathcal{X}_f^c$ and $\mathcal{X}_f^{c'}$.

Now, we can gather evidence from both the entity's own data and his/her neighbors' data to deduce the future events of the entity. For different rules, the selected entity-entity relations can be quite different. For example, if we want to infer the appearance of a disease in a patient. For contagious diseases, the entity-entity relation defined by the close contact will play a role. For inheritable diseases, such as hypertension, the parental relations will be taken into account.

Mathematically, given above rule template, we want to build the intensity function. One can introduce a *formula effect* (FE) to gather only the effective (i.e., nonzero) combinations of the historical predicate events as evidence to reason about the intensity of the head predicate for entity c. Introduce the logic function $g(\cdot)$ for the body conditions, which will be grounded by the effective (i.e., nonzero) combinations of the historical events of entity c and his/her neighbours.

$$\textit{Formula Effect:} \quad g\left(\{t_u^c\}_{u \in U}, \{t_v^{c'}\}_{v \in V}\right) \in \{0, 1\} \tag{5}$$

FE will be aggregated to reason about the intensity of the head predicate $Y(c, t_y)$ for entity c at any real time t_y. To ease the notation, define the predicate index set of the set of predicates $\mathcal{X}_f^c$ for entity c as U, and V for c'. Compute the feature as

$$\phi_f^c(t) = \sum_{c' \in \mathcal{C}} \sum_{\substack{\{\{t_u^c\} \in \mathcal{H}_t^{c,u}\}_{u \in U} \\ \{\{t_v^{c'}\} \in \mathcal{H}_t^{c',v}\}_{v \in V}}} R_d(c, c') g\left(\{t_u^c\}_{u \in U}, \{t_v^{c'}\}_{v \in V}\right) \tag{6}$$

where $\mathcal{H}_t^{c,u}$ is denoted as the historical trajectory specific to predicate u of entity c up to t. Here, given the temporal logic rule, we are gathering evidence from both the the entity's data and neighbors' data to build the feature. The information aggregation is realized by summation. This is intuitive. For example, for contagious diseases such as coronavirus, if more of a person's close contact exhibits symptoms or gets confirmed, this person's affected likelihood will be boosted. For inheritable diseases such as hypertension, if more of a person's parents or siblings get hypertension, this person's chance of getting hypertension should also be increased. In this paper, we also propose a unified TLPP rule learning framework that can automatically identify the possible entity-entity relations for each rule.

Suppose there is a rule set $\mathcal{F}$ can be used to deduce the head predicate Y. For each $f \in \mathcal{F}$, one can compute the features $\phi_f(t)$ as above, and assumes the rules are connected in disjunctive normal form (OR-of-ANDs). Given the feature constructed from each rule f, the intensity of the event process $\{Y(c,t)\}_{t\geq 0}$ can be modeled as monotonically increasing and non-negative functions of the weighted sum of the features:

$$\textit{Intensity: } \lambda^c(t \mid \mathcal{H}_t) = \exp\left(b_0 + \sum\nolimits_{f\in\mathcal{F}} w_f \cdot \phi_f^c(t)\right) \tag{7}$$

where $\boldsymbol{w} = [w_f]_{f\in\mathcal{F}} \geq 0$ are the learnable weight parameters associated with each rule, and b_0 is the learnable base intensity. All the model parameters can be learned by MLE where the likelihood can be written as a function of the intensity function as Eq. (1), i.e.,

$$\textit{Likelihood: } \prod_{c\in\mathcal{C}} \prod_{t_i^c \in \mathcal{H}_t} \lambda^c(t_i^c|\mathcal{H}_{t_i}) \cdot \exp\left(-\int_0^t \lambda^c(\tau|\mathcal{H}_\tau)d\tau\right). \tag{8}$$

4. Overall Learning Framework

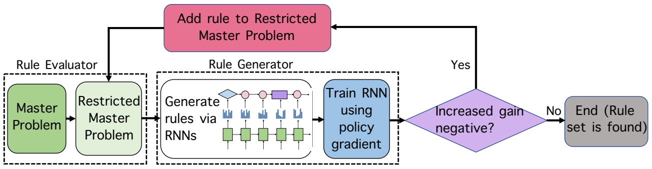


Figure 2. Overall Learning Framework: alternating process between rule generator and rule evaluator

Our goal is to jointly learn the model structure (i.e., rule set) and the model parameters by the MLE method that maximizes Eq.(1). It is common to assume that the hypothesized logic rules are in disjunctive normal form (DNF, OR-of-ANDs). In our problem, we assume each rule has an aforementioned general template, with a relatively complex structure. To discover each rule, the algorithm needs to navigate through the prespecified libraries of the entity-entity relation types, property predicates, and temporal relation constraints, and consider each possible combination. Moreover, each rule can have various lengths. Our learning problem is computationally challenging in nature. To address this issue, we first formulate the learning problem as a regularized convex problem. Then the learning problem is split into a restricted master problem (rule evaluator) and a subproblem (rule generator) and alternates between them, as demonstrated in Fig. 2. New rules are added in an incremental iterative way. Our overall learning framework is gradient-based. At each iteration of the subproblem, a gradient-based heuristic is used to quickly assess which rule candidate can most likely improve the existing model, and include it in the rule set. The restricted master problem will continue to train the model parameters using gradient descent (or SGD) to further improve the model. This two-stage algorithm terminates until by adding new rules the model cannot be improved. Similar ideas have been used in machine learning, including solving high-dimensional linear models (Perkins et al., 2003), mixed integer programming (Savelsbergh, 2002; Nemhauser, 2012; Lübbecke & Desrosiers, 2005), and large linear programming (Demiriz et al., 2002). Recently this method has been used to learn ordinal logic rules (Dash et al., 2018; Wei et al., 2019). Compared to these earlier works, our main difference is we propose a customized RL algorithm to solve the subproblem, which will further tackle the scalability issues. We will elaborate more on the proposed RL algorithm in the next section.

4.1. A Regularized Convex Problem

We first formulate the model learning problem as a regularized convex problem (the log-likelihood is known to be convex (Fahrmeir et al., 1994)), where the objective function is the log-likelihood and a rule complexity penalty, i.e.,

$$\begin{aligned} \textit{Orignial Problem}: \quad & \boldsymbol{w}^*, b_0^* = \underset{\boldsymbol{w}, b_0}{\operatorname{argmin}} -\ell(\boldsymbol{w}, b_0) + \Omega(\boldsymbol{w}) \\ & s.t. \quad w_f \geq 0, \quad f \in \bar{\mathcal{F}} \end{aligned} \tag{9}$$

where $\bar{\mathcal{F}}$ is the complete rule set, and $\Omega(\boldsymbol{w})$ is a convex regularization function that has a high value for "complex" rule sets. For example, we can formulate $\Omega(\boldsymbol{w}) = \lambda_0 \sum_{f\in\bar{\mathcal{F}}} c_f w_f$ where c_f is the rule length.

4.2. Restricted Master Problem

The original problem is hard to solve, due to that the set of variables is exponentially large and can not be optimized simultaneously in a tractable way. We therefore start with a restricted master problem (RMP), where the search space is much smaller. For example, we can start with an empty rule set, denoted as $\mathcal{F}_0 \subset \bar{\mathcal{F}}$. Then we gradually expand this subset to improve the results, this will produce a nested sequence of subsets $\mathcal{F}_0 \subset \mathcal{F}_1 \subset \cdots \subset \mathcal{F}_k \subset \cdots$. For each $\mathcal{F}_k$, $k = 0, 1, \ldots$, the restricted master problem is formulated by replacing the complete rule set $\bar{\mathcal{F}}$ with $\mathcal{F}_k$:

$$\begin{aligned} \textit{RMP}: \ \boldsymbol{w}_{(k)}^*, b_{0,(k)}^* = \underset{\boldsymbol{w}, b_0}{\operatorname{argmin}} & \ -\ell(\boldsymbol{w}, b_0) + \Omega(\boldsymbol{w}) \\ & s.t. \ w_f \geq 0, \ f \in \mathcal{F}_k. \end{aligned} \tag{10}$$

Solving the RMP corresponds to the evaluation of the current candidate rules. All rules in the current set will be reweighed. The optimality of the current solution can be verified under the principle of the *complementary slackness* for convex problems, which in fact leads to the objective function of our subproblem. More proof can be found in the Appendix.

4.3. Subproblem

A gradient-based subproblem is formulated to propose a new temporal logic rule, which can potentially improve the optimal value of the RMP most. Given the current solution $\boldsymbol{w}^*_{(k)}, b^*_{0,(k)}$ for the restricted master problem (10), a subproblem is formulated to minimize the increased gain:

$$\textit{Subproblem:}\quad \min_{\phi_f} \; -\frac{\partial \ell\left(\boldsymbol{w}, b_0\right)}{\partial w_f} + \frac{\partial \Omega(\boldsymbol{w})}{\partial w_f}\bigg|_{\boldsymbol{w}^*_{(k)}, b^*_{0,(k)}} \tag{11}$$

where ϕ_f is a rule-informed feature. We aim to search for a new rule so that the corresponding feature minimizes the above objective function. However, this search procedure is also computationally expensive. In the next section, we will discuss how to solve the subproblem more efficiently using reinforcement learning.

5. Subproblem: Learning to Generate a Temporal Logic Rule via Reinforcement Learning

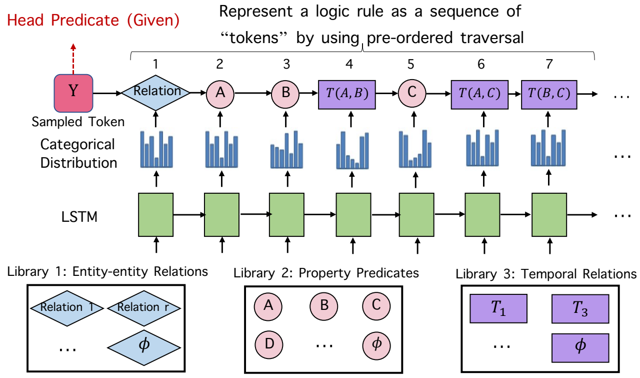

Head Predicate (Given)
Represent a logic rule as a sequence of "tokens" by using pre-ordered traversal
1
2
3
4
5
6
7
Y
Relation
A
B
$T(A,B)$
C
$T(A,C)$
$T(B,C)$
...
Sampled Token
Categorical Distribution
LSTM
Library 1: Entity-entity Relations
Relation 1
Relation r
...
ϕ
Library 2: Property Predicates
A
B
C
D
...
ϕ
Library 3: Temporal Relations
T_1
T_3
...
ϕ

Figure 3. Example of sampling a sequence from the LSTM

The subproblem formulated as Eq. (11) is to propose a new temporal logic rule. The subproblem itself is a minimization, which aims to attain the most negative increased gain. However, explicitly solving the subproblem requires enumerating all possible conjunctions of the input entity-entity relations, property predicates, and all possible pairwise temporal relations among the selected predicates, which is extremely computationally expensive. In this paper, instead of trying to enumerate all the conjunctions, we propose to learn a neural policy by reinforcement learning to generate the best-explanatory rules.

5.1. Generating Rules With Recurrent Neural Network

We leverage the fact that the temporal logic rules can be represented as a sequence of "tokens" subject to some unique structures. Using the pre-ordered traversal trick, we parameterize the policy by an RNN or LSTM, combined with dynamic masks, to guarantee that the generated tokens can yield a valid temporal logic rule.

We generate the rules one token at a time according to the pre-order traversal, as demonstrated in Fig. 3. Specifically, denote s as the state, which is the embedding of the previously generated tokens. We model the policy as $\pi_\theta(a|s)$ with learnable parameter θ, which quantifies the token selection probability given s. Each token/action can be chosen from the three predefined libraries: *i)* L_1, the entity-entity relation libraries, *ii)* L_2, then property predicate libraries, and *iii)* L_3, the temporal relation libraries.

Given the head predicate, we first generate the possible entity-entity relation token from L_1. If a non-empty token is generated, the search space of the following property predicates will be doubled, since the property predicates of the entity and his neighbor will be treated as different predicates in the rule. Every time a property predicate is generated, we need to consider its temporal relation with all the previously generated property predicates. Note that the temporal relation token can be None, which means there is no temporal relation constraints. All these generative prior knowledge can be incorporated as constraints by designing dynamic masks.

5.2. Policy gradient

The policy $\pi_\theta(a|s)$ is trained end-to-end by minimizing the subproblem objective using policy gradient, i.e.,

$$\max_\theta J(\theta) = \max_\theta \mathbb{E}_{\tau \sim \pi_\theta(\tau)}[R(\tau)]$$

where τ is the generated tokens (i.e., candidate rule) and

$$R(\tau) = \frac{\partial \ell\left(\boldsymbol{w}, b_0\right)}{\partial w_f} - \frac{\partial \Omega(\boldsymbol{w})}{\partial w_f}\bigg|_{\boldsymbol{w}^*_{(k)}, b^*_{0,(k)}}$$

The policy gradient can be estimated using the roll-out samples, i.e.,

$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum\nolimits_{i=1}^{N} \sum\nolimits_{k=1}^{K} \nabla_\theta \log \pi_\theta(a_k^{(i)} | s_k^{(i)}) R^{(i)}(\tau) \tag{12}$$

where N is the number of episodes, and K is the length of tokens (actions). We use this estimated policy gradient to update the policy θ, $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$, where α is the learning rate.

The proposed method has several benefits. First, we directly model the predicate selection probability, bypassing the enumerating of the discrete variables. The learning process of the neural guided policy is differential. Second, the neural policy is expressive and it is easy to transfer. The predicate selection priors can be refined after iterations and such knowledge can be encoded to the LSTM parameters, which can be reused for different subproblem iterations or even for different rule learning tasks. In addition, domain knowledge such as the rule templates can be easily incorporated as constraints by designing a dynamic mask, which can be applied to the predicate selection distributions at each step.

6. Experiments

6.1. Synthetic Data

We prepared 6 groups of event data, each group with a different set of ground truth rules. For all the datasets, we considered 20 entities connected by a random graph. The random graph was generated by an edge creation probability 0.3.

Different signal-noise ratio and scalability For each group, we further considered 4 cases, with the to-be-searched property predicate library being sized 8, 16, 24, and 32, respectively. Note that only a small amount of the predicates will be in the true rules. Many of the predicates are redundant information and they will act as background predicates. We aim to test whether our RL-based learning algorithm can truly uncover the rules from the noisy variable set and how the performance will evolve if we gradually increase the variable set with more and more redundant variables.

Different ground truth rule structures The designed ground truth rules of different groups are with various content structures. For some groups, each rule shares many common predicates in content, while for some groups, the designed rules are quite distinct in content. For example, the ground truth rules in group-$\{1, 2, 3\}$ are quite different in their content, and the ground truth rules in group-$\{4, 5, 6\}$ share many common predicates. We listed all the discovered rules for Group-1: Case 1, and Group-4: Case 1, in Tab. 1.

Table 1. *Ground truth rules which are learned and *Ground truth rules which are failed to learn.

Group-1 Case-1	Group-4 Case-1
*D(user) ← Relation(user, neighbor) ∧ C(user) ∧ A(neighbor) ∧ C(user) After A(neighbor)	*D(user) ← Relation(user, neighbor) ∧ C(user) ∧ A(neighbor) ∧ C(user) After A(neighbor)
*D(user) ← Relation(user, neighbor) ∧ B(user) ∧ B(neighbor) ∧ B(user) After B(neighbor)	*D(user) ← Relation(user, neighbor) ∧ C(user) ∧ B(neighbor) ∧ C(user) After B(neighbor)
*D(user) ← Relation(user, neighbor) ∧ A(user) ∧ D(neighbor) ∧ A(user) After D(neighbor)	*D(user) ← Relation(user, neighbor) ∧ C(user) ∧ D(neighbor) ∧ C(user) After D(neighbor)

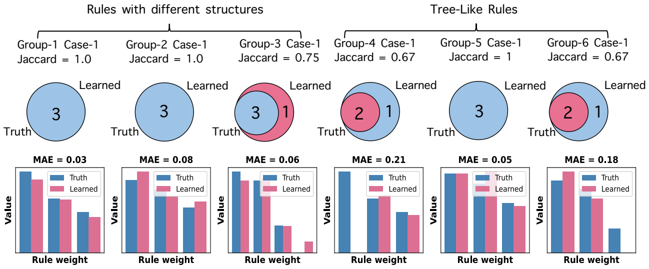


Figure 4. Rule discovery ability and rule weight learning accuracy of our proposed model based on dataset-1 for all 6 groups. Blue one indicates ground truth rule and red one indicates learned rule.

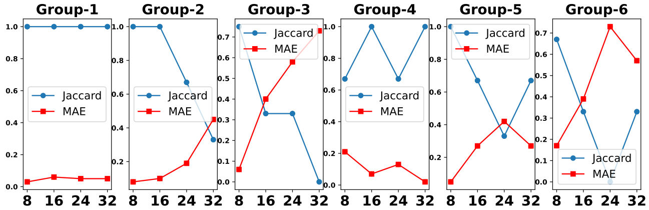


Figure 5. Jaccard similarity score and MAE for all 6 groups. X-axis indicates the predicate library size and Y-axis indicates the value of Jaccard similarity and MAE

In Fig. 4, we reported the learning results for the case with predicate size 8 for all groups using 1000 samples. Complete results for all the datasets can be found in Appendix. 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. Our proposed model discovered almost all the ground truth 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 the truth rule weights are accurately learned and the MAEs are quite low. In group-3 and group-6, we crafted long and complex rules with 4 body property predicates and various temporal relations, which yield an extremely huge search space, but our model still discovered almost all the ground truth rules.

Fig. 5 illustrates the Jaccard similarity score and MAE for all 6 groups using 1000 samples. For all 6 groups, as the number of predicates in the predicate set increases, the Jaccard similarity scores decrease slightly and the MAE increases slightly, but it is still within an acceptable range. This is because as the number of redundant predicates increases, the search space expands exponentially and the complexity of searching is dramatically increased. But if the number of predicates in the predicate set is appropriate and the samples are sufficient, our model is very stable and reliable.

It's worth noting that when training datasets in group-$\{1, 2, 3\}$, every time we enter the subproblem to search a candidate rule, we reset our LSTM model, aiming to remove the memory of the LSTM, since the ground truth rules in these groups are different in predicates and structure. We want to search totally different rules for these groups. But for group-$\{4, 5, 6\}$, the ground truth rules are similar in content. We want to search similar rules and the memory of the

LSTM will help in this case. So every time we enter subproblem, we still use the previously well trained LSTM to utilize its memory. By doing so, training time will be significantly reduced and our model is verified to be transferable for tree-like rules.

6.2. Healthcare Dataset

Table 2. Learned Rules with `LowUrine` as the head predicate.

Weight	**Rule**
0.4305	**Rule 1:**LowUrine ← LowWBC ∧ (LowWBC Before LowUrine)
0.0726	**Rule 2:**LowUrine ← HighBUN ∧ (HighBUN Before LowUrine)
0.0597	**Rule 3:**LowUrine← HighBUN ∧ HighArterialLactate ∧ (HighBUN Before LowUrine) ∧ (HighArterialLactate Before LowUrine)
1.2631	**Rule 4:**LowUrine ← LowArterialPh ∧ HighBUN ∧ (HighBUN Before LowUrine) ∧ (LowArterialPh Before LowUrine)
0.4914	**Rule 5:**LowUrine ← LowArterialPh ∧ Highspo2sao2 ∧ HighBUN ∧ (LowArterialPh Equal LowUrine) ∧ (Highspo2sao2 Equal LowUrine) ∧ (HighBUN Equal LowUrine)

MIMIC-III is an electronic health record dataset of patients admitted to the intensive care unit (ICU) (Johnson et al., 2016). We considered patients diagnosed with sepsis (Saria, 2018; Raghu et al., 2017; Peng et al., 2018), since sepsis is one of the major causes of mortality in ICU. Previous studies suggests that the treatment strategy is still unclear. It is unknown what is the optimal treatment strategy in terms of using 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 for the patients. We only use the proposed RL-based learning algorithm to learn explainable rules and their weights to shed light on this problem.

Table 3. Learned Rules with `NormalUrine` as the head predicate.

Weight	**Rule**
3.0325	**Rule 1:**NormalUrine ← NormalSysBP ∧ (NormalSysBP Before NormalUrine)
0.1986	**Rule 2:**NormalUrine ← NormalArterial ∧ (NormalArterial Before NormalUrine)
0.2620	**Rule 3:**NormalUrine ← NormalPotassium ∧ (NormalPotassium Before NormalUrine)
0.2982	**Rule 4:**NormalUrine ← ORcrystalloid ∧ (ORcrystalloid Before NormalUrine)
0.001	**Rule 5:**NormalUrine ← LowSpo2Sao2 ∧ LowCRP ∧ OrColloid ∧ (LowCRP Before NormalUrine) ∧ (LowSpo2Sao2 Before NormalUrine) ∧ (OrColloid Before NormalUrine)

Discovered temporal logic rules. Tab. 2 and 3 show the uncovered explanatory temporal logic rules and weights learned by our algorithm. We use `LowUrine` and `NormalUrine` as the head predicates respectively. We invited human experts (doctors) to check the correctness of these discovered rules and the doctors think these rules have scientific meaning and are consistent with the pathogenesis of sepsis.

Table 4. MIMIC-III: Event prediction results.

Method	LowUrine (ACC)	NormalUrine (ACC)
RMTPP	0.713	0.757
NHP	0.724	0.763
TR-GRU	0.813	0.803
LSTM	0.808	0.796
Transformer	0.819	0.814
OURMETHOD	**0.831**	**0.873**

We consider the following SOTA baselines: 1) Recurrent Marked Temporal Point Processes (RMTPP) (Du et al., 2016), 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, 2017), an improved variant of RMTPP by constructing a continuous-time LSTM; 3) Tree-Regularized GRU (TR-GRU) (Wu et al., 2018), a deep time-series model with a designed tree regularizer to add model interpretability; 4) Long Short-Term Memory (LSTM) (Staudemeyer, 2019), an advanced RNN, a sequential network, that allows information to persist; 5) Transformer (Zuo et al., 2020), an advanced model which follows an encoder-decoder structure, but does not rely on recurrence and convolutions in order to generate an output.

We use accuracy (ACC) as the evaluation metrics for the state prediction tasks of the two head predicates: 1) `LowUrine` and 2) `NormalUrine`, which are evaluated by predicting the time of transitions from state 0 to state 1. Larger ACC indicates better performance of model. The performance of our model and all baselines are compared in Tab. 4, from which one can observe that our model outperforms all the baselines in this experiment.

7. Conclusion

In this paper, we proposed a unified temporal logic point process modeling framework for event data. We further designed a learning algorithm that automates the rule discovery process from data to explain the event data. The learning algorithm alternates between a rule generator stage and a rule evaluator stage, where a neural guided policy is introduced to generate candidate rules. The use of a neural policy improves the transferability. We evaluate our methods on both synthetic and healthcare datasets, obtaining promising results.

Acknowledgements

Thanks to Shuang Li for useful guidance, feedback and discussion. This work was in part supported by her Start-up Fund UDF01002191 of The Chinese University of Hong Kong, Shenzhen, Shenzhen Institute of Artificial Intelligence and Robotics for Society, and Shenzhen Science and Technology Program JCYJ20210324120011032.

References

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

Dash, S., Gunluk, O., and Wei, D. Boolean decision rules via column generation. *Advances in neural information processing systems*, 31, 2018.

Demiriz, A., Bennett, K. P., and Shawe-Taylor, J. Linear programming boosting via column generation. *Machine Learning*, 46(1):225–254, 2002.

Du, N., Dai, H., Trivedi, R., Upadhyay, U., Gomez-Rodriguez, M., and Song, L. 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, 2016.

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

Fahrmeir, L., Tutz, G., Hennevogl, W., and Salem, E. *Multivariate statistical modelling based on generalized linear models*, volume 425. Springer, 1994.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. Soft actor-critic: off-policy maximum entropy deep reinforcement learning with a stochastic actor.

Johnson, A. E., Pollard, T. J., Shen, L., Li-Wei, H. L., Feng, M., Ghassemi, M., Moody, B., Szolovits, P., Celi, L. A., and Mark, R. G. Mimic-iii, a freely accessible critical care database. *Scientific data*, 3(1):1–9, 2016.

Landajuela, Petersen, K. S. G. N. M. P. F. Discovering symbolic policies with deep reinforcement learning. pp. 5979–5989, 2021.

Li, S., Wang, L., Zhang, R., Chang, X., Liu, X., Xie, Y., Qi, Y., and Song, L. Temporal logic point processes. In *International Conference on Machine Learning*, pp. 5990–6000. PMLR, 2020.

Li, S., Feng, M., Wang, L., Essofi, A., Cao, Y., Yan, J., and Song, L. Explaining point processes by learning interpretable temporal logic rules. In *International Conference on Learning Representations*, 2021.

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

Mei, H. and Eisner, J. M. The neural hawkes process: A neurally self-modulating multivariate point process. *Advances in neural information processing systems*, 30, 2017.

Mohler, G. O., Short, M. B., Brantingham, P. J., Schoenberg, F. P., and Tita, G. E. Self-exciting point process modeling of crime. *Journal of the American Statistical Association*, 106(493):100–108, 2011.

Nemhauser, G. L. Column generation for linear and integer programming. *Optimization Stories*, 20(65):U73, 2012.

Peng, X., Ding, Y., Wihl, D., Gottesman, O., Komorowski, M., Lehman, L.-w. H., Ross, A., Faisal, A., and Doshi-Velez, F. 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.

Perkins, S., Lacker, K., and Theiler, J. Grafting: Fast, incremental feature selection by gradient descent in function space. *The Journal of Machine Learning Research*, 3: 1333–1356, 2003.

Petersen, Larma, N. M. S. K. T. Deep symbolic regression: recovering mathematical expressions from data via risk-seeking policy gradients. 2021.

Raghu, A., Komorowski, M., Ahmed, I., Celi, L., Szolovits, P., and Ghassemi, M. Deep reinforcement learning for sepsis treatment. *arXiv preprint arXiv:1711.09602*, 2017.

Rasmussen, J. G. Lecture notes: Temporal point processes and the conditional intensity function. *arXiv preprint arXiv:1806.00221*, 2018.

Sadeghian, A., Armandpour, M., Ding, P., and Wang, D. Z. Drum: End-to-end differentiable rule mining on knowledge graphs. *arXiv preprint arXiv:1911.00055*, 2019.

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

Savelsbergh, M. W. Branch-and-price: Integer programming with column generation, bp. 2002.

Staudemeyer, R. M. Understanding lstm – a tutorial into long short-term memory recurrent neural networks. *arXiv preprint arXiv:1909.09586*, 2019.

Wang, P.-W., Donti, P., Wilder, B., and Kolter, Z. Satnet: Bridging deep learning and logical reasoning using a differentiable satisfiability solver. In *International Conference on Machine Learning*, pp. 6545–6554. PMLR, 2019a.

Wang, P.-W., Stepanova, D., Domokos, C., and Kolter, J. Z. Differentiable learning of numerical rules in knowledge graphs. In *International Conference on Learning Representations*, 2019b.

Wang, T., Rudin, C., Doshi-Velez, F., Liu, Y., Klampfl, E., and MacNeille, P. A bayesian framework for learning rule sets for interpretable classification. *The Journal of Machine Learning Research*, 18(1):2357–2393, 2017.

Wei, D., Dash, S., Gao, T., and Gunluk, O. Generalized linear rule models. In *International Conference on Machine Learning*, pp. 6687–6696. PMLR, 2019.

Wu, M., Hughes, M., Parbhoo, S., Zazzi, M., Roth, V., and Doshi-Velez, F. Beyond sparsity: Tree regularization of deep models for interpretability. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

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

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

Zhang, Q., Lipani, A., Kirnap, O., and Yilmaz, E. Self-attentive hawkes process. In *International conference on machine learning*, pp. 11183–11193. PMLR, 2020a.

Zhang, W., Panum, T., Jha, S., Chalasani, P., and Page, D. Cause: Learning granger causality from event sequences using attribution methods. In *International Conference on Machine Learning*, pp. 11235–11245. PMLR, 2020b.

Zhu, S., Ng, I., and Chen, Z. Causal discovery with reinforcement learning. 2020.

Zoph, B. and Le, Q. V. Neural architecture search with reinforcement learning. *International Conference on Machine Learning*, 2017.

Zuo, S., Jiang, H., Li, Z., Zhao, T., and Zha, H. Transformer hawkes process. In *International conference on machine learning*, pp. 11692–11702. PMLR, 2020.

Appendix Overview

In the following, we will provide supplementary materials to better illustrate our methods and experiments.

- Section A is about our related works.
- Section B provides the definitions of all types of temporal relations considered in our model.
- Section C and D elaborate on the necessary proofs, which justify our model and learning framework.
- Section E provides pseudocode to illustrate our rule generator policy, subproblem formulation, and the complete algorithm.
- Section F provides our computing infrastructure.
- Section G, H and I show the comprehensive experiments on synthetic data. In I we further considered a risk-seeking policy gradient in solving the subproblem and we compared the performance to the standard policy gradient.
- Section J introduces the background on the MIMIC-III data and provides a list of the chosen predicates used in our real experiment.

A. Related Work

Temporal point process (TPP) models. TPP models can be characterized by the intensity function. The modeling framework boils down to the design of various intensity functions to add the flexibility and interpretability of the model (Mohler et al., 2011). Recent development in deep learning has significantly enhanced the flexibility of TPP models. (Du et al., 2016) proposed a neural point process model, named RMTPP, where the intensity function is modeled by a Recurrent Neural Network. (Mei & Eisner, 2017) improved RMTPP by constructing a continuous-time RNN. (Zuo et al., 2020) and (Zhang et al., 2020a) recently leveraged the self-attention mechanism to capture the long-term dependencies of events. Although flexible, these neural TPP models are black-box and are hard to interpret. To add transparency, (Zhang et al., 2020b) introduced Granger causality as a latent graph to explain point processes, and the structures are jointly learned via gradient descent. However, Granger causality is still limited to the triggering patterns of events. Recently, (Li et al., 2020) proposed an explainable Temporal Logic Point Process (TLPP), where the intensity function is informed by a set of temporal logic rules. TLPP model enables one to perform symbolic reasoning based on TPP and a follow-up work (Li et al., 2021) designed a branch and price algorithm to learn the set of explanatory logic rules and their weights. By contrast, our model considered a more general and complicated temporal logic rule template and proposed a reinforcement learning algorithm to learn the rules.

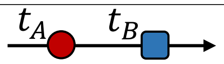
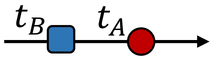
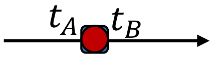
Logic rule learning methods. To learn the ordinary logic rules without temporal relation constraints, previous works addressed the problem from various perspectives. For example, (Evans & Grefenstette, 2018; Wang et al., 2017) formulated the logic learning problem as learning an explanatory binary classifier, where the latter one is a Bayesian framework. SATNet (Wang et al., 2019a) transformed rule mining into a SDP-relaxed MaxSAT problem. Attention-based methods (Yang & Song, 2019) were also introduced. Neural-LP (Yang et al., 2017) provided the first fully differentiable rule mining method based on TensorLog (Cohen, 2016), and (Wang et al., 2019b) extended Neural-LP to learn rules with numerical values via dynamic programming and cumulative sum operations. In addition, DRUM (Sadeghian et al., 2019) connected learning rule confidence scores with low-rank tensor approximation. Recently, (Dash et al., 2018; Wei et al., 2019) introduced a column generation (i.e., branch and price) type of MIP algorithm to learn the logic rules using different objective functions. However, all these above-mentioned methods are restricted to static data without timestamps and cannot be directly implemented on sequential data or event sequences. In (Li et al., 2021), a column generation algorithm was formulated to learn the temporal logic rules based on the TPP models via maximizing the likelihood. In this paper, as a comparison, we adopted the similar column generation idea to learn the temporal logic rules in a progressive way, however, we proposed an efficient reinforcement learning algorithm to solve the subproblem. We generated new candidate rules by a trained neural guided policy, where this is realized by enumerating in (Li et al., 2021).

Learning model structures via reinforcement learning (RL). Recently, RL provided a promising approach to automatically finding the best-fitting model structures. For example, in AutoML, the famous NAS (Zoph & Le, 2017) trained a recurrent neural network by RL to design the architectures of deep neural networks and has achieved comparable performance with the human-designed models. Similar ideas have been adopted to aid the design of the explainable machine learning models. For example, the RL algorithm has been successfully introduced to learn the causal graph to explain the data (Zhu et al., 2020). Recently, the RL algorithm has been used in symbolic regression (Petersen, 2021; Landajuela, 2021), which aims to learn the set of compact mathematical expressions to explain the dynamics of complex dynamic systems. In our paper, we introduced and customized the RL algorithm to learn the temporal logic rules to explain the event sequences.

B. Detailed Explanation of Temporal Relation

In this paper, the temporal relation was defined among events. For any pairwise events, denoted as A and B, there exist only three types of temporal relations, which can be grounded by their occurrence times, denoted as t_A and t_B. See below Table 5 for illustrations.

Table 5. Event-based temporal relations.

Temporal Relation	Mathematical Expression	Illustration
A Before B	$t_A < t_B$	
A After B	$t_A > t_B$	
A Equals B	$t_A = t_B$	

The temporal relation of any two events will be treated as temporal ordering constraints and can be included in a temporal logic rule as Eq. (4). Note that when included in the rule, the temporal relation can be none, which indicates that there is no temporal relation constraint between the two events in order to satisfy the rule.

C. Proof of The Likelihood Function of TLPP

The likelihood function of the TLPP is a straightforward result from TPP. Readers can refer to the proofs in (Rasmussen, 2018). To be self-contained, we will provide a sketch of proof here.

For a specific entity c, given all the events associated with the head predicate $(t_1^c, t_2^c, \dots) \in [0, t)$, the likelihood function is the joint density function of these events. Using the chain rule, the joint likelihood can be factorized into the conditional densities of each points given all points before it. For entity c, this yields:

$$\mathcal{L}^c = p^c\left(t_1^c \mid \mathcal{H}_0\right) p^c\left(t_2^c \mid \mathcal{H}_{t_1^c}\right) \cdots p^c\left(t_n^c \mid \mathcal{H}_{t_{n-1}^c}\right)\left(1 - F^c\left(t \mid \mathcal{H}_{t_n^c}\right)\right) \tag{13}$$

where $p^c(t \mid \mathcal{H}_{t_n})$ represents the conditional density and $F^c(t \mid \mathcal{H}_{t_n})$ refers to its cumulative distribution function for any $t > t_n$. $(1 - F^c\left(t \mid \mathcal{H}_{t_n}\right))$ appears in the likelihood since the unobserved point t_{n+1} hasn't happened up to t. Further, by the hazard rate definition of the intensity function

$$\lambda^c(t) = \frac{p^c\left(t \mid \mathcal{H}_{t_n}\right)}{1 - F^c\left(t \mid \mathcal{H}_{t_n}\right)} \tag{14}$$

we will have

$$p^c(t|\mathcal{H}_{t_n}) = \lambda^c(t) \exp\left(-\int_{t_n}^{t} \lambda^c(s) ds\right). \tag{15}$$

Using the above equation, we can get

$$\begin{aligned}
\mathcal{L}^c &= \left(\prod_{i=1}^{n} p^c\left(t_i^c \mid \mathcal{H}_{t_i^c-1}\right)\right)\frac{\lambda^c(t)}{p^c(t \mid \mathcal{H}_{t_n})} \\
&= \left(\prod_{i=1}^{n} \lambda^c(t_i^c)\exp\left(-\int_{t_{i-1}^c}^{t_i^c} \lambda^c(\tau)d\tau\right)\right)\exp\left(-\int_{t_n^c}^{t} \lambda^c(\tau)d\tau\right) \\
&= \left(\prod_{i=1}^{n} \lambda^c(t_i^c)\right)\exp\left(-\int_{0}^{t} \lambda^c(\tau)d\tau\right).
\end{aligned} \tag{16}$$

Now consider the likelihood function of all entities $\mathcal{C} = \{c_1, c_2, ..., c_n\}$, which can be factorized according to the entities, the likelihood can be written as

$$\textit{Likelihood:} \quad \prod_{c\in\mathcal{C}}\prod_{t_i^c\in\mathcal{H}_t} \lambda^c(t_i^c|\mathcal{H}_{t_i})\cdot\exp\left(-\int_0^t \lambda^c(\tau|\mathcal{H}_\tau)d\tau\right) \tag{17}$$

which completes the proof.

D. Optimality Condition and Complementary Slackness

We will provide more descriptions on the optimality condition and the complementary slackness, which provides a sound guarantee to our learning algorithm.

Given the original restricted convex problem,

$$\textit{Orignial Problem}: \quad \boldsymbol{w}^*, b_0^* = \underset{\boldsymbol{w}, b_0}{\operatorname{argmin}} -\ell(\boldsymbol{w}, b_0) + \Omega(\boldsymbol{w}); \quad s.t. \quad w_f \geq 0, \quad f \in \bar{\mathcal{F}} \tag{18}$$

where $\Omega(\boldsymbol{w})$ is a *convex* regularization function that has a high value for "complex" rule sets. For example, we can formulate $\Omega(\boldsymbol{w}) = \lambda_0 \sum_{f\in\bar{\mathcal{F}}} c_f w_f$ where c_f is the rule length.

The Lagrangian of the original master problem is

$$L(\boldsymbol{w}, b_0\boldsymbol{\nu}) = -\ell\left(\boldsymbol{w}, b_0\right) + \Omega(\boldsymbol{w}) - \sum_{f\in\bar{\mathcal{F}}} \nu_f w_f, \tag{19}$$

where $\nu_f \geq 0$ is the Lagrange multiplier associated with the non-negativity constraints of w_f. As it is a convex problem and strong duality holds under mild conditions. Define $\boldsymbol{w}^*, b_0^*$ as the primal optimal, and $\boldsymbol{\nu}^*$ as the dual optimal, then:

$$\begin{aligned}
-\ell(\boldsymbol{w}^*, b_0^*) &= \inf_{\boldsymbol{w}, b_0} \; L(\boldsymbol{w}, b_0, \boldsymbol{\nu}^*) \qquad \text{(strong duality)} \\
&= \inf_{\boldsymbol{w}, b_0}\left(-\ell\left(\boldsymbol{w}, b_0\right) + \Omega(\boldsymbol{w}) - \sum_{f\in\bar{\mathcal{F}}} \nu_f^* w_f\right) \\
&\leq -\ell\left(\boldsymbol{w}^*, b_0^*\right) + \Omega(\boldsymbol{w}^*) - \sum_{f\in\bar{\mathcal{F}}} \nu_f^* w_f^* \\
&\leq -\ell\left(\boldsymbol{w}^*, b_0^*\right) + \Omega(\boldsymbol{w}^*).
\end{aligned} \tag{20}$$

Therefore, $\sum_{f\in\bar{\mathcal{F}}} \nu_f^* w_f^* = 0$, for $f \in \bar{\mathcal{F}}$. This implies the *complementary slackness*, i.e.,

$$w_f^* = 0 \Rightarrow \nu_f^* \geq 0, \qquad w_f^* > 0 \Rightarrow \nu_f^* = 0 \tag{21}$$

Given the Karush-Kuhn-Tucker (KKT) conditions, the gradient of Lagrangian $L(\boldsymbol{w}^*, b_0^*, \boldsymbol{\nu}^*)$ w.r.t. $\boldsymbol{w}^*, b_0^*$ vanishes, i.e.,

$$\nu_f^* := -\left.\frac{\partial\left[\ell(\boldsymbol{w}, b_0) - \Omega(\boldsymbol{w})\right]}{\partial w_f}\right|_{\boldsymbol{w}^*, b_0^*}. \tag{22}$$

In summary, combining conditions (21) and (22), we obtain the optimalitiy condition of the original problem,

1. if $w_f^* > 0$, then $\nu_f^* = 0$;
2. if $w_f^* = 0$, then $\nu_f^* \geq 0$,

where the gradient ν_f^* can be computed via (22). At each iteration, we solve the subproblem to find the candidate rule that most violates this optimality condition, i.e., yields the most negative Eq. (22).

E. Algorithm Box

Our method alternates between solving a restricted master problem and a subproblem. When executing the subproblem, we need to generate several candidate rules. We summarize the algorithm in Algorithm 1, Algorithm 2, and Algorithm 3. RG refers to the Rule Generator used to generate a new candidate rule when solving the subproblem. Here, we will parameterize the RG as a LSTM. SP is the abbreviation of Sub-Problem which is optimized to construct a new rule. RMP indicates the Restricted Master Problem used to update model parameters.

```
Algorithm 1: Rule Generator (RG)
Input: RuleLen, HeadPred
Output: CandidateRule
RelationLibrary ← $\{R_1, R_2, \ldots, NotR\}$; // $R_d$ indicates different entity-entity relation
    types.  If NotR is chosen, it implies there will be no relation predicate
    defined in the rule and all the evidence should come from the entity's own
    history.
PredLibrary_1 ← $\{A(user), B(user), \ldots\}$;
PredLibrary_2 ← $\{A(user), B(user), \ldots, A(neighbor), B(neighbor), \ldots\}$; // Search space doubled if
    $R_d$ is chosen.
TempRelationLibrary← $\{Before, After, Equal, None\}$;
Sequence ← EmptySet;
BodyPredSet ← EmptySet;
Relation ← LSTM(HeadPred, RelationLibrary);
Sequence.add(Relation);

if Relation is NotR then
    PredLibrary ← PredLibrary_1
else
    PredLibrary ← PredLibrary_2

DynamicMask = All-Ones

while BodyPredNum≤ RuleLen do
    NewBodyPred ← LSTM(Sequence, PredLibrary ⊙ DynamicMask)
    DynamicMask ← Set the location of NewBodyPred zero (DynamicMask)
    Sequence.add(NewBodyPred);

    for BodyPred in BodyPredSet do
        // We need to consider the pairwise temporal relation between the new
            selected body predicate and all the body predicates that already have
            been selected in the body predicate set.
        TempRelation ← LSTM(BodyPred, NewBodyPred, TempRelationLibrary);
        Sequence.add(TempRelation);

    BodyPredSet.add(NewBodyPred);

CandicateRule ← Convertor(Sequence); // Convert a sequence of tokens to rule template.

return CandicateRule
```

Algorithm 2: Sub-Problem (SP)

```
Input: RuleLen, HeadPred
Output: NewRule

while $iter \leq Total\_Iter$ do
    CandidateRuleBatch ← EmptySet

    while $Batch\_idx \leq BatchSize$ do
        CandidateRule ← RG(RuleLen, HeadPred); // Generate candidate rule.
        CandidateRuleBatch.add(CandidateRule);

    Policy ← Policy.update// Policy gradient

    if $PolicyGradientNorm \leq threshold$ then
        FinalCandidateRule ← RG(RuleLen, HeadPred);
        return FinalCandidateRule

FinalCandidateRule ← RG(RuleLen, HeadPred);
return FinalCandidateRule
```

Algorithm 3: Complete model

```
Input: RuleLen, HeadPred, TotalRuleNum
Output: ruleSet

ruleSet ← EmptySet;
$\boldsymbol{b} \leftarrow \mathbf{0}$;
$\boldsymbol{w} \leftarrow \mathbf{0}$;
$\boldsymbol{b}, \boldsymbol{w}$ ← RMP($\boldsymbol{b}, \boldsymbol{w}$, ruleSet); // Initialize weights and base terms.

while $CurrentRuleNum \leq TotalRuleNum$ do
    NewRule ← SP(RuleLen, HeadPred); // Generate candidate rule.

    if $IncreasedGain<0$ then
        ruleSet.add(NewRule);
        $\boldsymbol{b}, \boldsymbol{w}$ ← RMP($\boldsymbol{b}, \boldsymbol{w}$, ruleSet); // After adding new rule, update weights and base
            terms

return $\boldsymbol{b}, \boldsymbol{w}$, ruleSet
```

F. Computing Infrastructure.

All experiments are performed on Ubuntu 20.04 LTS system with Intel(R) Xeon(R) CPU E5-2690 v3 @ 2.60GHz CPU, 112 Gigabyte memory and single NVIDIA Tesla P100 accelerator.

G. Complete Results of Experiments on Synthetic Data

Fig. 6 and Fig. 7 demonstrate the learning results for the cases with predicate size 8, 16, 24, and 32 (predicates in the library that we need to search) for all groups using 1000 samples of networked events. We set the learning rate in solving the subproblem to be $\times 10^{-2}$. Hidden state size of LSTM was 32. The learning rate in solving the restricted master problem was $\times 10^{-3}$. Our proposed model discovered almost all the ground truth rules for all cases in all groups. And almost all the truth rule weights are accurately learned and the MAEs are quite low. For cases in group-3 and group-6, we considered long and complex rules with 4 body property predicates and the associated temporal relations, which yield a very big search space, but our model still discovered most of the ground truth rules.

We also evaluated our model using 500 and 2000 samples with the same experiment settings. Our model also achieved satisfactory performance when sample size is small, and the performance of the model was further improved as the sample size increases.

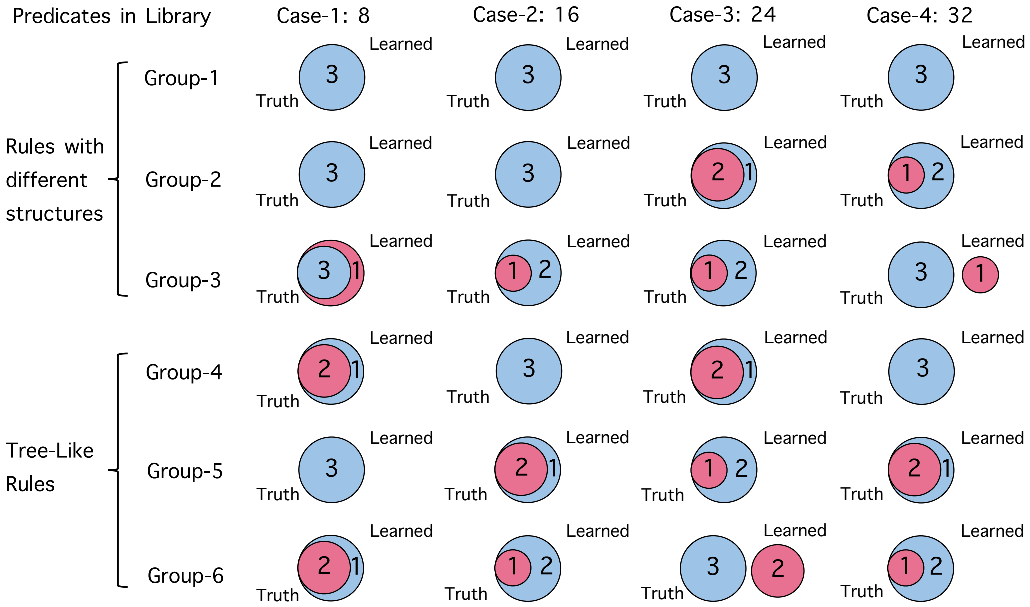


Figure 6. Jaccard similarity score for all 6 groups and all 4 cases using 1000 samples. Blue one indicates the ground truth rule and red one indicates the learned rule.

H. Reuse the LSTM Memory and Early Stop Mechanism

When to reuse the LSTM memory for different subproblem iterations? The ground truth rules in group-$\{1, 2, 3\}$ are quite distinct in their content with almost no common predicates in each rule. Given this prior knowledge, when training the datasets in these groups, whenever the algorithm enters the subproblem to search for a new candidate rule, we reset our LSTM model and clean the memory. If not, we will get the convergence results as illustrated in Fig. 8, where we observe that keeping the LSTM memory will hinder the convergence speed of the subproblems. As illustrated in Fig. 9 (left) where we refresh the LSTM model parameters whenever the algorithm enters the subproblem stage. As a comparison, by doing this, the convergence of the subproblem is much faster. The ground truth rules in group-$\{4, 5, 6\}$ are similar in content and share many predicates. Given this prior knowledge, reuse the LSTM across subproblems may help the convergence.

Early stop mechanism To speed up our algorithm, we propose an early stop mechanism. We consider that when the iteration times reach a pre-set reasonable number, and when the LSTM model consecutively generates identical logic rules

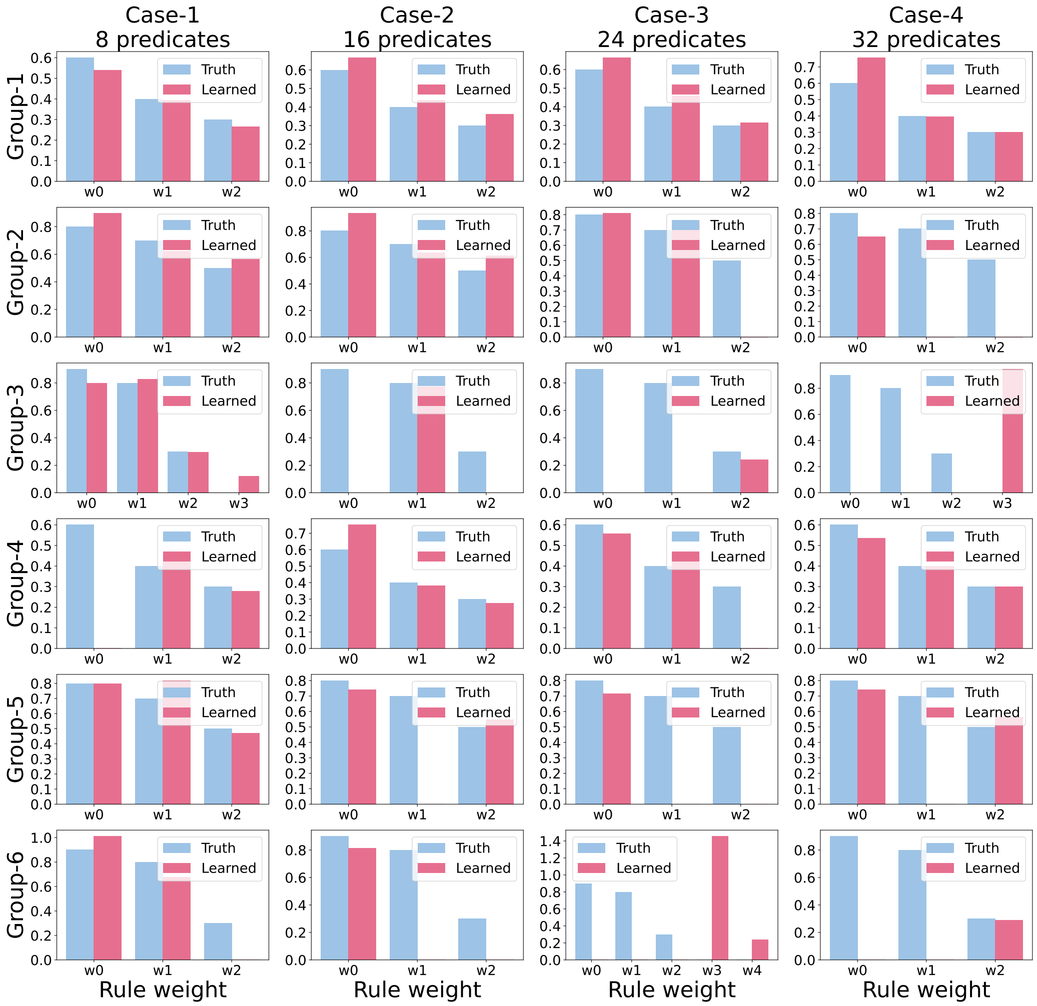


Figure 7. MAE values for all 6 groups and all 4 cases using 1000 samples. Blue one indicates the ground truth rule and red one indicates the learned rule.

(like identical 10 rules), we conclude that the LSTM model has been trained enough and is able to generate a good-performing logic rule. We don't need to train the LSTM until the norm of policy gradient is within a very small tolerance. Hence, we may early stop our training. Fig. 9 (right) illustrates the training trajectories of solving subproblem for group-1 case-1, where we reset the LSTM in the beginning of the subproblems and use early stop mechanism. It's obvious that this mechanism can reduce redundant iterations and help us complete the training process in a short time.

I. Risk-Seeking Policy Gradient

The standard policy gradient objective $J(\theta)$ is defined as an expectation. This is the desired objective for control problems in wihch one seeks to optimize the average performance of a policy. However, rule learning problems described in our paper are to search for best-fitting rules. For such problems, $J(\theta)$ may not appropriate, as there is a mismatch between the objective being optimized and the final performance metric.

We consider risk-seeking policy gradient like (Petersen, 2021; Landajuela, 2021), which proposed an alternative objective that aims to **maximize the best-case performance**. According to the original work (Landajuela, 2021; Petersen, 2021), we first define $R_\epsilon(\theta)$ as the $(1-\epsilon)$-quantile of the distribution of rewards under the current policy. Then the new objective $J_{risk}(\theta;\epsilon)$ is given by:

$$J_{risk}(\theta;\epsilon) = \mathbb{E}_{\tau\sim\pi(\tau|\theta)}\left[R(\tau) \mid R(\tau) \geq R_\epsilon(\theta)\right] \tag{23}$$

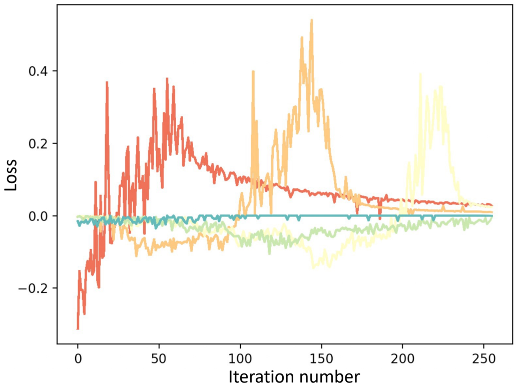


Figure 8. Training loss trajectories of solving different subproblem stages if the LSTM model parameters are inherited from the previous subproblem.

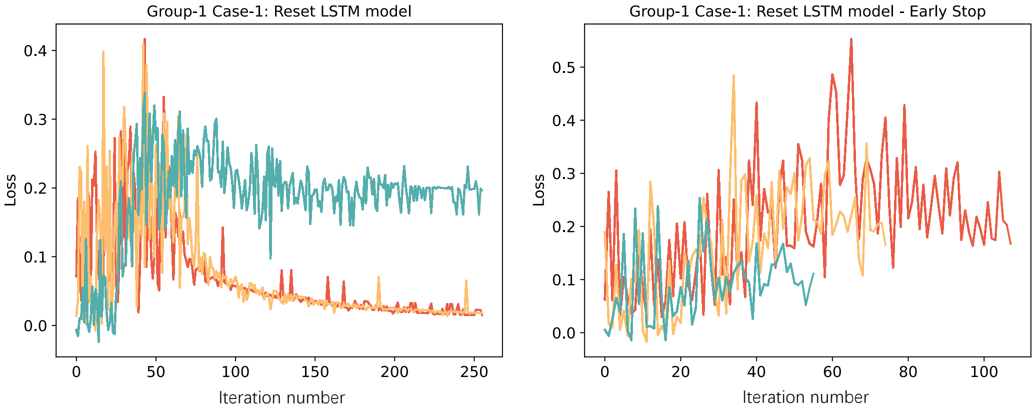


Figure 9. Training loss trajectories of solving different subproblem stages for group-1 case-1. Left: The LSTM parameters are refreshed in the beginning of each subproblem. Right: Early stop mechanism is used.

Then the risk-seeking policy gradient can be estimated using the roll-out samples, i.e.,

$$\nabla_\theta J_{risk}(\theta;\epsilon) \approx \frac{1}{\epsilon N}\sum_{i=1}^{N}\sum_{k=1}^{K}\left[R^{(i)}(\tau)-\tilde{R}^{(i)}_{\epsilon}(\theta)\right]\cdot \mathbb{1}_{R^{(i)}(\tau)\geq \tilde{R}^{(i)}_{\epsilon}(\theta)}\nabla_\theta \log \pi_\theta(a_k^{(i)}|s_k^{(i)}) \tag{24}$$

where N is the number of episodes, and K is the length of tokens (actions). $\tilde{R}^{(i)}_{\epsilon}(\theta)$ is the empirical $(1-\epsilon)$-quantile of the batch of rewards, and $\mathbb{1}$ returns 1 if condition is true and 0 otherwise. We use this estimated policy gradient to update the policy θ.

$$\theta \leftarrow \theta + \alpha\nabla_\theta J_{risk}(\theta;\epsilon) \tag{25}$$

where α is the learning rate.

Further, according to the maximum entropy reinforcement learning framework (Haarnoja et al.), a bonus can be added to the loss function proportional to the entropy to help the policy do the exploration.

We notice that the performance of our model is less satisfactory for cases in group-3 and group-6, mainly because the ground truth rules are long and complex with many body property predicates and various temporal relations. So we redid the experiments but using risk-seeking policy gradient to see whether there are some improvements. We set ϵ to be 0.3. The learning rate in solving the subproblem was $\times 10^{-2}$ and the hidden state size of LSTM was 32. The learning rate in solving the restricted master problem was $\times 10^{-3}$. The results are shown in Fig. 10 and Fig. 11.

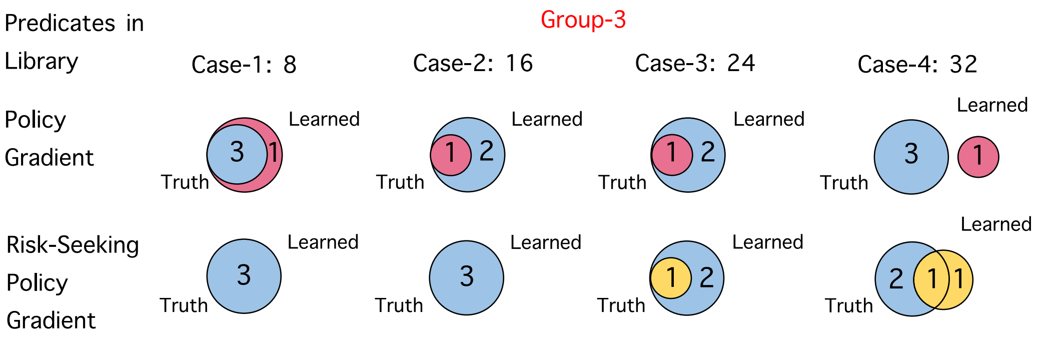


Figure 10. Jaccard similarity score for all 4 cases in group-3 using 1000 samples. Blue one indicates the ground truth rule, red one indicates the learned rule using policy gradient, and yellow one indicates the learned rule using risk-seeking policy gradient.

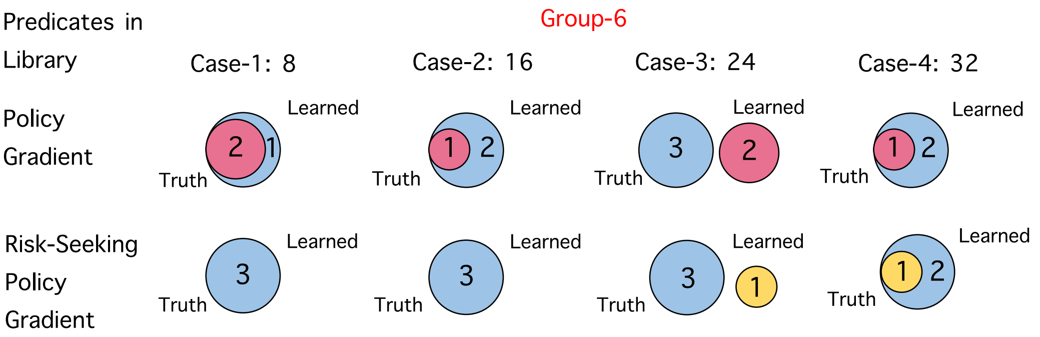


Figure 11. Jaccard similarity score for all 4 cases in group-6 using 1000 samples.

The results show that more ground truth rules were learned by the risk-seeking policy gradient, which can significantly improve the performance of our model.

J. Predicate Definition in MIMIC-III

MIMIC-III is an electronic health record ICU dataset, which is released under the PhysioNet Credentialed Health Data License 1.5.0[1]. It was approved by the Institutional Review Boards of Beth Israel Deaconess Medical Center (Boston, MA) and the Massachusetts Institute of Technology (Cambridge, MA). In this dataset, all the patient health information was deidentified. We manually checked that this data do not contain personally identifiable information or offensive content.

We defined 63 predicates, including two groups of drugs (i.e., intravenous fluids and vasopressors) and lab measurements, see Tab. 6 for more details. Among all these predicates we were interested in reasoning about two predicates and define them as head predicates: 1) `LowUrine` and 2) `NormalUrine`. We treated real time urine as head predicates since low urine is the direct indicator of bad circulatory systems and the signal for septic shock; normal urine reflects the effect of the drugs and treatments and the improvement of the patients physical condition. In our experiments, lab measurement variables were converted to binary values (according to the normal range used in medicine) with the transition time recorded. For drug predicates, they were recorded as 1 when they were applied to patient. We extracted 2023 patient sequences, and randomly selected 80% of them for training and the remaining 20% for testing. The average time horizon is 392.69 hours and the average events per sequence is 79.03.

[1] https://physionet.org/content/mimiciii/view-license/1.4/

Table 6. Defined Predicates in Our MIMIC-III Experiment.

Lab Measurements	Low/Normal/High-SysBP Low/Normal/High-SpO2SaO2 Low/Normal/High-CVP Low/Normal/High-SVR Low/Normal/High-Potassium Low/Normal/High-Sodium Low/Normal/High-Chloride Low/Normal/High-BUN Low/Normal/High-Creatinine Low/Normal/High-CRP Low/Normal/High-RBCcount Low/Normal/High-WBCcount Low/Normal/High-ArterialpH Low/Normal/High-ArterialBE Low/Normal/High-ArterialLactete Low/Normal/High-HCO3 Low/Normal/High-SvO2ScvO2
Output	LowUrine, NormalUrine
Input	Colloid, Crystalloid, Water
Drugs	Norepinephrine, Epinephrine, Dobutamine, Dopamine, Phenylephrine
Temporal Relation Type	Before, After, Equal