Debiased Imitation Learning for Modulated Temporal Point Processes

Zhuoqun Li[*] Zihan Zhou[†] Mingxuan Sun[‡] Hongteng Xu[§]

Abstract

Temporal event sequences associated with different event types (e.g., location indices, disease types) are observed in various applications such as disaster resilience, criminology, and healthcare. Temporal point processes (TPPs) have been developed to capture the exciting patterns between events and forecast future events quantitatively. Unfortunately, the events with different types often suffer from unknown biased observations in real-world scenarios due to external interference. Accordingly, the temporal point processes learned by conventional maximum likelihood estimation (MLE) from such biased data may be misspecified and may lead to inaccurate predictions. To overcome this issue, we model biased event sequences as modulating TPPs with additional unknown thinning processes. Furthermore, we develop a novel debiased imitation learning framework to learn the modulated TPPs and suppress the negative influences of biased data, which is more robust than conventional MLE. When applying the debiased imitation learning framework, we design a simple but effective reward function based on the historical embedding obtained by the TPP model. Experiments on three real-world datasets demonstrate that our proposed method significantly outperforms existing methods.

Keywords: Imitation Learning; Temporal Point Process.

1 Introduction

Temporal event sequences associated with different event types (e.g., location indices, disease types, tweet categories) are observed in various applications such as disaster resilience, criminology, healthcare, and social media. The events in such sequences often exhibit strong self- and mutually-exciting patterns, e.g., a crime event is likely to trigger more crimes in nearby areas and time slots, an alert of disasters like floods may trigger a series of posts and forwards on social media, and a TV viewing event may trigger the users' visit to remaining episodes daily or weekly in the following days.

Temporal point processes (TPPs) like Hawkes process (HP) [1] and its neural network-based variants [2, 3] have been developed to capture the exciting patterns between events and forecast future events quantitatively. For example, in a predictive rescue system during flood disasters, a city is divided into geographic sub-regions such as grid cells or political boundaries. Each "S.O.S." event is recorded with a time stamp and an event type (e.g., a zip code). A TPP model is used to predict the rates of rescue events for each region at each time window based on historical data. Regions are then ranked by the predicted hazard rates in each time window so that rescue activities are directed to top-k risky regions, also known as hotspot detection [1].

In real-world event forecasting, events with different types and timestamps often suffer from imperfect observations due to external interference. A typical example of this phenomenon is the "S.O.S." events of flood disasters collected from social media or sensor networks, where events associated with certain types (e.g., zip codes) may occur but are unobserved due to power outages or infrastructure failures in these regions. Besides the biased sampling of specific event types, the events in specific time windows may have a lower or higher sampling probability than others. For example, the dynamic of flood sensor networks is time-varying.

Such imperfect observations introduce undesired "bias" to the estimation of TPP models — for the maximum likelihood estimation of a TPP based on such data, the asymptotic unbiasedness of the mean intensity may not be held.[1] Accordingly, TPP models learned by conventional maximum likelihood estimation (MLE) from such biased data may be misspecified and may lead to biased predictions in the testing phase when external interference is alleviated. For example, a TPP model is used to forecast the rates of crime events for each region in a city based on historical training data. Events reported in different regions may be biased in the training data due to the fact that more police resources are allocated to some regions than the others. However, in the testing time window, more resources are available and police patrol activities are uniform across the city. In such cases, a TPP model trained on crime data with biased observations would fail to predict events accurately in the test phase.

Existing efforts have been made to learn TPPs from imperfect observations, e.g., the time-varying Hawkes process with stitching-based augmentations [4], the Hawkes process with hidden types [5], and a bidirectional continuous-time LSTM [6]. However, they focus on modeling missing event types or time windows, rather than explicitly modeling the events with unknown biased generative mechanism. Moreover, they do not solve the mismatching problem between the distribution of training and the testing data caused by the sampling bias of external interference.

In this paper, we propose a novel algorithmic frame-

[*]zli82@lsu.edu, Louisiana State University
[†]clairechou1112@gmail.com, Cognomotiv
[‡]msun11@lsu.edu, Louisiana State University
[§]hongtengxu313@gmail.com, Renmin University of China

[1]In this study, the "bias" is defined for the estimation of the mean intensity of point process, which will be explained in detail in Section 2.2.

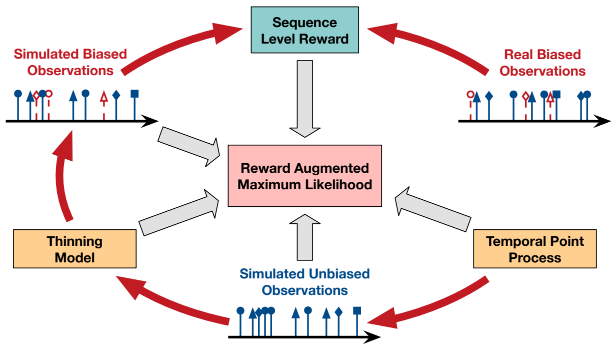

Sequence Level Reward
Simulated Biased Observations
Real Biased Observations
Reward Augmented Maximum Likelihood
Thinning Model
Temporal Point Process
Simulated Unbiased Observations

Figure 1: The scheme of proposed learning method. The red dotted stems represent the events generated by the system but unobserved because of the additional thinning modulated on the target TPP.

work, called “debiased imitation learning (DIL)”, to learn temporal point processes from biased observations and suppress the risk of model misspecification accordingly. Specifically, we model the mechanism of the biased observation explicitly, modulating the target TPP model by an additional biased thinning process of event types. Unlike other modulated methods such as [2,7], this thinning process is external, which not only generates biased sequences but also imposes interference on the real triggering patterns among the events. Our DIL framework learns the TPP and the external thinning process jointly with the help of the reward-augmented maximum likelihood (RAML) [8].

As illustrated in Fig. 1, our framework imitates the generative mechanism of the biased data based on current model parameters, and then measures the discrepancy between the real and the simulated biased sequences in an efficient and effective method. Using the discrepancy to construct a sequence-level reward, our DIL learns model parameters by policy gradient-based updating. Experimental results demonstrate that our DIL framework is applicable to arbitrary modulated TPPs and suppresses the negative influences of biased data, which is more robust than conventional maximum likelihood estimation (MLE) and its variants. Also, the learned TPP model predicts further events without bias, which performs more fairly across different event types in the testing phase.

2 Proposed Model

2.1 Temporal point processes Denote a temporal point process (TPP) with C event types as $\boldsymbol{N}(t) = [N_c(t)]$. Its realization in a time window $[0, T)$ is an event sequence consisting of a list of events, *i.e.*, $s = \{(t_i, c_i)\}_{i=1}^N$ with $0 < t_1 < \cdot < t_n < T$ and $c_i \in \mathcal{C} = \{1, 2, .., C\}$ for $i = 1, ..., N$. For the i-th event (t_i, c_i), t_i and c_i represents the event timestamp and type, respectively. As aforementioned, many real-world sequential data can be described by the above temporal point process. For example, in crime and disaster rescue event sequences [1,9], each event happens at a specific time and is associated with a ZIP code (*i.e.*, event type).

Typically, a TPP model is characterized by conditional intensity functions $\boldsymbol{\lambda}(t) = \{\lambda_c(t)\}_{c\in\mathcal{C},t\in[0,T)}$, where each $\lambda_c(t)$ is the expected instantaneous rate of the type-c event at time t given all historical events up to time t, i.e.,

$$\lambda_c(t|\mathcal{H}_t^{\mathcal{C}}) = \frac{\mathbb{E}[\mathrm{d}N_c(t)|\mathcal{H}_t^{\mathcal{C}}]}{\mathrm{d}t}, \quad \forall c \in \mathcal{C}, t \in [0, T), \tag{2.1}$$

where $N_c(t)$ counts the number of the type-c events till time t and the history $\mathcal{H}_t^{\mathcal{C}} = \{(t_i, c_i)|t_i < t, c_i \in \mathcal{C}\}$.

The formulation of the above intensity functions determines the exciting patterns among the events, and thus, leads to various TPP models. For example, the homogeneous Poisson process [10] is memoryless, which owns constant intensity functions, i.e., $\lambda_c(t) = \eta_c \geq 0$, where η_c is a positive constant associated with event type c. The classic Hawkes process [11] applies additive intensity functions, i.e.,

$$\lambda_c(t|\mathcal{H}_t^{\mathcal{C}}) = \mu_c + \sum_{i:t_i<t} \phi_{c,c_i}(t - t_i), \tag{2.2}$$

which captures the influence of the type-c' event at time t' on the type-c event at time t by an impact function $\phi_{cc'}(t - t_i)$ and accumulates all historical influences in an additive way. Recently, neural network-based Hawkes processes leverage more sophisticated mechanisms to quantify the influence of historical events, whose intensity functions can be represented in general as

$$\lambda_c(t|\mathcal{H}_t^{\mathcal{C}}) = g_c(t, h(\mathcal{H}_t^{C})), \tag{2.3}$$

where h is a neural network embedding the historical events to vectorized hidden state, which can be a recurrent neural network [3], a continuous-time LSTM [2], or a transformer [12], and g_c is the following neural network corresponding to the type-c event, which takes time and the embedding as its inputs.

Typically, given a set of K sequences in $[0, T)$, denoted as $\mathcal{S} = \{s_k\}_{k=1}^K$, the likelihood of observing the sequences given a TPP model parameterized by θ is:

$$L(\theta; \mathcal{S}) = \prod_{k=1}^{K} \frac{\prod_{i=1}^{N_k} \lambda_{c_i^k}(t_i^k|\mathcal{H}_{t_i^k}^{\mathcal{C}}; \theta)}{\exp(\sum_{c\in\mathcal{C}} \int_0^T \lambda_c(\tau|\mathcal{H}_\tau^{\mathcal{C}}; \theta)d\tau)}, \tag{2.4}$$

where the intensity functions λ are parameterized by θ. We can learn a TPP model by the maximum likelihood estimation (MLE):

$$\min_\theta -\log L(\theta; \mathcal{S}). \tag{2.5}$$

2.2 The bias of mean intensity caused by external interference It should be noted that the intensity functions are random variables depending on historical observations. For a multivariate temporal point process $\boldsymbol{N}(t)$, whose intensity functions are $\{\lambda_c(t)\}_{c\in\mathcal{C}}$, the mean intensity of each event type is defined as the expectation of $\lambda_c(t)$ over all history [13], i.e.,

$$\bar{\lambda}_c = \mathbb{E}_{\mathcal{H}_t^{\mathcal{C}}\sim\mathcal{P}}[\lambda_c(t|\mathcal{H}_t^{\mathcal{C}})],\ \forall c\in\mathcal{C}, \tag{2.6}$$

where $\mathcal{P}$ indicates a probability measure of possible history. The mean intensity reflects the overall dynamics of a point process model, which is determined by the model parameters in general, i.e., $\bar{\lambda}_c(\theta)$ for $c\in\mathcal{C}$. For typical point processes like Poisson process and Hawkes process, the mean intensity has a closed form solution [13].

Suppose that we observe the sequences $\mathcal{S}\sim\boldsymbol{N}_{\theta^*}(t)$ in $[0,T)$, where $\boldsymbol{N}_{\theta^*}(t)$ is a TPP with ground truth parameter θ^*. The MLE is proven to provide a consistent estimation of the TPP [14], and accordingly, the mean intensity functions own asymptotically unbiased estimations, i.e., $\mathbb{E}_{\hat{\theta}}[\bar{\lambda}_c(\hat{\theta})]\rightarrow\bar{\lambda}_c(\theta^*)$ when $T\rightarrow\infty$.

However, as aforementioned, practical event sequences often suffer from external interference. In such a situation, the observed sequences $\mathcal{S}$ are actually from a TPP modulated by unknown noise. Accordingly, the probability measure of history, i.e., the $\mathcal{P}$ in (2.6) is changed. When applying the MLE on the noisy data $\mathcal{S}$, the model has a high risk of misspecification, and the asymptotic unbiasness of the mean intensity cannot be held. To solve this problem, we need to take the external interference into account in the modeling phase and develop new debiased learning algorithms accordingly.

2.3 Modulating TPPs via additional thinning As aforementioned, in many real-world applications, events with different types often suffer from unknown biased observations — the events might be *generated by the system but unobserved because of external interference*. A typical example of this phenomenon is the "S.O.S." events of flood disasters. Many "S.O.S." events actually happened but are not recorded by media or rescue systems because the infrastructures of disaster areas, like power grids and telecommunication networks, are destroyed by floods. As a result, learning a TPP from the observed data directly will leads to a serious model misspecification issue and generates biased disaster forecasting and evaluation for different areas.

To overcome this issue, we first propose the following generative mechanism to generate above biased observations. Suppose that we have a TPP model $\boldsymbol{N}(t)$, which generates "unobservable" unbiased event sequences. The observed biased event sequence is generated by modulating the TPP with an additional thinning process, denoted as $\{m_c\}_{c\in\mathcal{C}}$, which remove the generated type-c events with rate m_c. We can formulate this generative process as:

$$\begin{aligned}&\text{1.Simulate unbiased sequences (Ogata's thinning):}\\&s=\{(t_i,c_i)\}_{i=1}^N\sim\boldsymbol{N}(t).\\&\text{2.Additional thinning:}\\&\text{For } i=1,...,N:\\&\quad u\sim\text{Uniform}([0,1]), s\leftarrow s\setminus\{(t_i,c_i)\}\text{ if } u>m_{c_i}\\&\text{Biased sequences: } \hat{s}\leftarrow s.\end{aligned} \tag{2.7}$$

Here, the first step generates unbiased data from the target TPP, which can be achieved by Ogata's thinning algorithm [15]. The formulation of the additional thinning rates $\{m_c\}_{c\in\mathcal{C}}$ is flexible, which depends on practical application scenarios. Typically, we can set each m_c as a time-invariant rate for sampling type-c events. In such a situation, the target TPP is modulated via a noise multivariate Poisson process. In more general settings, m_c can be a function of time and even parameterized by neural networks. As a result, we obtain the biased data $\hat{s}$ by (2.7), and we aim at learning the target TPP model $\boldsymbol{N}(t)$ and the associated additional thinning process $\{m_c\}_{c\in\mathcal{C}}$ from the data.

It should be noted that the modulated temporal point process (MTPP) in (2.7) is different from those in [7, 16]. Specifically, conventional MTPPs modulates point processes by Markovian or Gaussian processes, in which the modulations are intrinsic mechanisms of the TPP models. In other words, when generating event sequences, the modulation happens during the Ogata's simulation step: given historical events, the modulation is a necessary part to determine current intensity, which ensures the whole simulation process to generate desired event sequences. On the contrary, the additional thinning imposed on our TPP model is an external interference, rather than a module of the target model itself, which works after the simulation step and leads to biased data.

A challenge caused by the proposed additional thinning process is making the MLE in (2.5) inapplicable. Specifically, denote $\mathcal{H}_t^{\mathcal{C}}$ as the historical events in the unbiased sequence s before time t. Similarly, we denote $\widehat{\mathcal{H}}_t^{\mathcal{C}}$ as the observed historical events in the biased sequence $\hat{s}$ before time t. The likelihood function in (2.4) consists of the intensity functions that take historical observations as their inputs. According to (2.7), we can only observe $\hat{s}$ and the corresponding $\widehat{\mathcal{H}}_t^{\mathcal{C}}$. As a result, for each event (t,c) in the biased sequence $\hat{s}$, we cannot obtain the intensity function $\lambda_c(t|\mathcal{H}_t^{\mathcal{C}};\theta)$ directly because the complete history $\mathcal{H}_t^{\mathcal{C}}$ is unavailable. On the other hand, estimating the intensity function from biased history, i.e., $\lambda_c(t|\widehat{\mathcal{H}}_t^{\mathcal{C}};\theta)$, will lead to a misspecified model.

As a result, the likelihood function based on the biased sequence, i.e., $L(\theta;\hat{s})$, is either unavailable (if considering $\lambda_c(t|\mathcal{H}_t^{\mathcal{C}};\theta)$) or misspecified (if considering $\lambda_c(t|\widehat{\mathcal{H}}_t^{\mathcal{C}};\theta)$). To overcome this challenge, we propose a new learning framework called debiased imitation learning in the next section, which provides a potential solution beyond the MLE strategy to learn unbiased TPPs from biased observations.

3 Learning Algorithm

In principle, our debiased imitation learning (DIL) framework learn the target TPP and the additional thinning process jointly with the help of the reward-augmented maximum likelihood (RAML) [8]. The proposed framework consists of a learner and an expert. The learner imitates the generation process in (2.7), generating unbiased and biased sequences based on current models. The expert evaluates the quality of the generated sequences by comparing them with real observations and assign different rewards to the generated sequences. The goal of our DIL is to find the optimal model that maximizes the expected reward.

3.1 Debiased imitation learning framework Suppose that we have an initial estimation of the TPP model θ and the additional thinning $\{m_c\}_{c\in\mathcal{C}}$, we can simulate N unbiased sequences and their biased observations by the steps in (2.7), denoted as $\{s_n, \tilde{s}_n\}_{n=1}^N$. For each $\tilde{s}_n$, its intensity function can be derived without bias because the corresponding unbiased sequence s_n is available. In particular, for each event $(t, c) \in \tilde{s}_n$, we can obtain its intensity $\lambda_c(t|\mathcal{H}_t^{\mathcal{C}};\theta)$, where $\mathcal{H}_t^{\mathcal{C}} \subset s_n$. Accordingly, the likelihood function based on $\tilde{s}_n$ becomes available, i.e.,

$$(3.8)\quad L(\theta,\{m_c\};\tilde{s}_n,s_n) = \frac{\prod_{(t_i^n,c_i^n)\in\tilde{s}_n}\lambda_{c_i^n}(t_i^n|\mathcal{H}_t^{\mathcal{C}};\theta)}{\exp(\sum_{c\in\mathcal{C}}\int_0^T m_c\lambda_c(\tau|\mathcal{H}_\tau^{\mathcal{C}};\theta)d\tau)}$$

Compared with the likelihood in (2.4), the likelihood in (3.8) owns two differences: i) for each event in the biased sequence $\tilde{s}_n$, its history is from the unbiased sequence s_n; ii) the integration in the denominator corresponds to the expected biased number of events, which take the additional thinning into account.

Based on the generated paired sequences, their likelihoods, and K real-world biased sequences $\{\hat{s}_k\}_{k=1}^K$, our DIL solves the following reward-augmented maximum likelihood problem:

$$(3.9)\quad \min_{\theta,\{m_c\}_{c\in C}} -\sum_{k=1}^K\sum_{n=1}^N r(\hat{s}_k,\tilde{s}_n)\log L(\theta,\{m_c\};\tilde{s}_n,s_n),$$

where $r(\hat{s}_k,\tilde{s}_n)$ is a sequence-level reward function measuring the similarity between the real sequence and the simulated sequence. The design of the reward function is critical for our learning method, which will be discussed in Section 3.2.

Given the rewards, the model parameter θ and the additional thinning $\{m_c\}$ are updated by an alternating optimization strategy based on policy gradients [17]:

$$(3.10)\quad \begin{aligned} \theta &\leftarrow \theta+\alpha\sum_{k,n} r(\hat{s}_k,\tilde{s}_n)\nabla_\theta \log L(\theta,\{m_c\};\tilde{s}_n,s_n),\\ m_c &\leftarrow m_c+\alpha\sum_{k,n} r(\hat{s}_k,\tilde{s}_n)\nabla_{m_c}\log L(\theta,\{m_c\};\tilde{s}_n,s_n)\end{aligned}$$

where α is learning rate, and the gradients above are calculated via backpropagation.

Note that, for the thinning process $\{m_c\}_{c\in\mathcal{C}}$, we can further parameterize it when some prior knowledge about the biased mechanism is available. For example, in the sequence of online rescue events, each event type (i.e., a ZIP code c) may be associated with a feature x_c including the information of power grids, network facility, and social-economic status of the corresponding region. Accordingly, we can model each m_c as a learnable function of x_c, such as $m_c = f(x_c)$ where f is a multi-layer perceptron (MLP). In summary, Algorithm 1 shows the scheme of our DIL framework in details.

Algorithm 1 Debiased Imitation Learning (DIL)

```
Biased training sequences $\{\hat{s}_k\}$;
Model parameters $\theta$, thinning rates $\{m_c\}_{c\in\mathcal{C}}$ Initialize $\theta$ and $\{m_c\}_{c\in\mathcal{C}}$ randomly;
i ← 1 to MaxIter:
    1. Simulated $N$ unbiased and biased sequences $\{s_n,\tilde{s}_n\}_{n=1}^N$ by (2.7) from current model $\theta$ and $\{m_c\}_{c\in\mathcal{C}}$.
    2. Calculate reward $r(\hat{s}_k,\tilde{s}_n)$.
    3. Update model parameter by (3.10).
```

It should be noted that the original RAML [8] is designed for discrete or categorical data, whose sample space is finite, while our DIL simulates unbiased event sequences in the continuous-time domain and thus the search space is infinite. To make our DIL feasible, in practice, we modify the Ogata's thinning algorithm, simulating sequences in a "finite" search space. In particular, following the idea of nonparametric point process models [18], we can discretize the time window as M bins, and each bin can contain at most one event. When applying the Ogata's thinning method to simulate event sequences, for the timestamp fall into a bin, we determine whether there exists an event in it and which type the event is. As a result, our search space becomes finite when ignoring the uncertainty of the timestamp within each bin, whose size is $(|\mathcal{C}|+1)^M$.

3.2 Proposed reward function As aforementioned, the proposed reward function should capture the similarity between arbitrary two sequences. A typical option is designing the reward as a kernel function based on the distance between the sequences, i.e., $r(s,s') = \exp(-d(s,s'))$, where the distance $d(s,s')$ can be the Wasserstein distance in [19]. The

Wasserstein distance computes the distance between events of the same type and average the distance over all types, and for each event type sequences are sorted and compared in order. Formally, given two sequences $\hat{s}_k$ and $\tilde{s}_n$ in $[0, T)$, let $\xi_c = \{t_1, t_2, .., t_n\}$ and $\rho_c = \{\tau_1, \tau_2, .., \tau_m\}$ be two sub sequences of event type c sorted by time stamps increasingly. Without loss of generality we assume $n \leq m$ and the distance between these two as define in [19] is:

$$d(\xi_c, \rho_c) = \sum_{i=1}^{n} |t_i - \tau_i| + \sum_{i=n+1}^{m} (T - \tau_i), \tag{3.11}$$

where the second summation is basically padding the shorter sequences with T. The reward between two sequences is thus:

$$r(\hat{s}_k, \tilde{s}_n) = \exp\left(-\frac{\gamma}{C}\sum_{i=1}^{C} d(\xi_c, \rho_c)\right), \tag{3.12}$$

where C is number of event types and γ is a positive constant.

Another popular choice for $d(s, s')$ is the hierarchical optimal transport (HOT) distance in [20]. The HOT distance further considers a hierarchical architecture for the sequences with different event types, which applies nested Wasserstein distance and thus owns higher complexity. Unfortunately, these distances are often computational expensive for comparing the event sequences with multiple event types. The kernel-based reward used in [21] is only applicable for spatio-temporal point process, rather than our TPPs with categorical event types. In summary, for the real-world scenarios like city crime and disaster rescue, whose event sequences often own hundreds of event types (i.e., ZIP codes), these distances are hard to scale and unsuitable for designing computationally-friendly reward functions.

In this work, we propose a simple but efficient reward function for the generalized TPP model defined in (2.3). Specifically, the TPP in (2.3) leverages a neural network h to obtain the embedding of historical events. The embedding obtained by h aggregates the information of the input events. When the input of h is the whole sequence, *i.e.*, $h(s)$, we can obtain an embedding that represents the sequence. Therefore, given $\hat{s}_k$ and $\tilde{s}_n$, we derive the reward $r(\hat{s}_k, \tilde{s}_n)$ as a function of the cosine similarity between their embeddings:

$$r(\hat{s}_k, \tilde{s}_n) = f\left(\frac{h^\top(\hat{s}_k)h(\tilde{s}_n)}{\|h(\hat{s}_k)\|_2\|h(\tilde{s}_n)\|_2}\right), \tag{3.13}$$

where f can be a simple scaling function or a nonlinear activation. This design is efficient because we can derive the reward easily by the feed-forward computation of the model. Compared with distance-based methods such as the Wasserstein distance, our reward function works well in the following experiments (Section 5) and reduces computational cost greatly.

4 Related Work

4.1 Temporal point processes Conditional point processes, such as Hawkes processes, are capable of modeling self- and mutual-excited event sequences and can be used for event clustering [22], causal inference [11], and event forecasting [1]. Point processes have been widely applied in various applications including predictive policing [1, 5] and online social activities [3, 23, 24]. In particular, those TPPs outperform regression models for predicting event hazard rates and ranking event hotspots [1]. In comparison with traditional Hawkes approaches, neural Hawkes processes such as recurrent marked temporal point process (RMTPP) [3], continuous-time LSTM [2], transformer Hawkes [12], and self-attentive Hawkes [25], show better prediction accuracy in event forecasting. Typically, the above TPP models are learned in the MLE framework, and few of them consider learning based on imperfect observations.

4.2 Learning from imperfect observations Some efforts have been made to learn TPPs from imperfect observations, e.g., applying data augmentation methods in the training phase [4, 6] and imposing structural (e.g., low-rank, sparse, and topological) regularizers on the models [26]. Another attempt [5] and its variant [9] assume that events of certain types may be unobserved during any time period, which can excite or be excited by events of observed types. A particle smoothing method based on a bidirectional continuous-time LSTM is proposed in [6] to simulate missing events for missing windows.

A neural network model is proposed in [27] to model both observed and missing sequences. However, most of those methods focus on modeling missing event types or time windows, rather than explicitly modeling the events with unknown biased generative mechanism. Moreover, they do not solve the mismatching problem between the training and the testing data caused by the biased observations. Some other approaches such as [1, 28] introduce fairness penalties into the likelihood function of spatial-temporal Hawkes processes to enforce group fairness in the hotspot predictions. While we focus on explicitly modeling data bias mechanism, those classic MLE frameworks with fairness regularization are complimentary to our approaches.

4.3 Reinforcement learning for TPPs Imitation learning and reinforcement learning have been used for learning TPPs [17], marked TPPs [29], and the clustering of temporal event sequences [22]. The primary benefit of those methods is that they do not rely on the pre-defined likelihood function and are more robust to model misspecification or overfitting than MLE. A recent imitation learning framework [21] has been specifically designed to estimate the parameters of a spatio-temporal Hawkes process given fully observed data in continuous space and time. A reward function is based on the

Table 1: Descriptions of real-world datasets.

Dataset	Events	Event Type	Unique-IDs	Time
IPTV	9230	TV ID	7	$200h$
Houston	1182	ZIP Code	106	$26h$
Dallas	3372	ZIP Code	74	$380d$

maximum mean discrepancy (MMD) and the discrepancy between sequences is computed via spatio-temporal kernels. Unlike [21], our framework focuses on modeling the event sequences with categorical types, which imitates biased generative processes in sequence simulation and is applicable to arbitrary modulated TPPs.

5 Experiment

5.1 Data We evaluate our methods on synthetic datasets as well as three real-world datasets. Our synthetic event streams are sampled based on Ogata's thinning algorithm from two different processes, respectively: (a) a multivariate Hawkes process [10] (M-SYN), (b) a neural Hawkes process (equivalently, a CT-LSTM model) [2] (N-SYN) with randomly initialized parameters. Sequences of $k = 5$ types are generated during the time window 0 hour to 35 hour and the length of each synthetic sequence L is from 100 events to 128 events. For each dataset, we generated 50 sequences. The M-SYN dataset contains 5867 events and the N-SYN dataset contains 5764 events in total.

The summary of real-world datasets is in Table 1. Specifically, the **IPTV** dataset [11] collects watching records of $7,100$ users and 436 TV programs in the period of January to November 2012. We select the records of 688 users watching 7 types of TV programs during 200 hours. There are totally $9,230$ events. The dataset **Houston** rescue data[2], which contains online rescue requests from Harris county in Houston area during the Hurricane Harvey. Each rescue request event has a time stamp and a ZIP code as the event type. There are $1,182$ events reported from 106 ZIP codes during 26 hours. The **Dallas** dataset, which is obtained from Kaggle[3], contains different types of crime incidences collected for around 3 years from the Dallas Police Department. We select "ROBBERY OF BUSINESS" events for a length of 380 days. Each robbery event has a time stamp and a ZIP code as the event type. There are totally 74 types.

5.2 Constructing biased training data To simulate real-world biased scenarios, i.e., training on biased data while testing on unbiased testing, we create biased training datasets with different sampling patterns from the original datasets. Specifically, we consider the original data as unbiased data and split it into a training set and a testset. Then we follow two different strategies to create biased training sets from the original training set. We train different models on the biased training sets and evaluate them on the unbiased testset.

The first strategy is "Random", where we randomly sample events from the training data. The sampling rate m_c for each type c is a random number ranging from 0 to 1. This strategy simulates the real-world scenarios where sampling bias exists at different locations due to random failure rates of sensor networks for spatial temporal events. The second strategy is "Top". Specifically, we rank event types by their total occurrences in the training data. For the most popular type, we keep all the events. For all other types, we randomly select 60% of the events. This strategy simulates the real-world scenarios where biased data with imbalanced distributions are observed such as online TV browsing events and hotspot events.

5.3 Baselines We consider the following learning methods including our DIL-based method and its competitors:

$\mathbf{MLE_H}$ is the method proposed in [5]. It adopts MLE to estimate a classic multivariate Hawkes process (MHP). In addition, it assumes that there exists hidden nodes (i.e., unobserved event types) that can excite or be excited by events of observed types. The method adopts a Metropolis-Hastings sampler to simulate virtual events to improve model estimation.

$\mathbf{MLE_R}$ is a method proposed in [9], which is an extension of $\mathbf{MLE_H}$. It defines a missing probability for each time interval and revises event likelihood function by integrating missing window probabilities. Similar to $\mathbf{MLE_H}$, a sampler is used to simulate virtual events. The paper also proposes to utilize prior knowledge of event types for model regularization. For a fair comparison, we choose the one without event type regularization as our baseline.

$\mathbf{MLE_{NN}}$ is a representative work that adopts MLE to estimate neural network (NN)-based point processes that are robust to biased cases, such as the continuous-time LSTM (CT-LSTM) in [2, 6]. Generally, these NN-based models learned by MLE outperform classic methods in prediction accuracy and robustness.

Our methods are denoted as $\mathbf{DIL_{NN}^W}$ and $\mathbf{DIL_{NN}^E}$, respectively. These two methods learn the aforementioned CT-LSTM model in the DIL framework, rather than the above MLE-based strategy, whose reward functions are different. $\mathbf{DIL_{NN}^W}$ uses traditional **W**asserstein distance [19] as defined in Equation (3.12) and $\mathbf{DIL_{NN}^E}$ uses our **E**mbedding-based distance defined in Equation (3.13).

5.4 Evaluation metrics We adopt different metrics to evaluate the models. Firstly, we use the data in the full time window and split it to train and test. The training data is further used to create biased training sets as described in Section 5.2. Model parameters are estimated using the biased

[2]https://data.world/sya/harvey-rescue-doc

[3]https://www.kaggle.com/carrie1/dallaspolicereportedincidents

training data and we evaluate the testset log-likelihood.

Furthermore, we would like to evaluate the accuracy of predicted event type c_i of the testing event at time t based on historical events $H(t_i)$. In this setting, events before time t are in the training set and events after time t are in the testset. The training data is further used to create biased training sets as described in Section 5.2. Model parameters are estimated using the biased training data. At each unique time unit t in the testset, the model can sample a set of events associated with different event types. We can rank the event types (e.g., hotspots) by the number of occurrences in the sampled sequences and compare the relevance of the ranked list with respect to the ground truth.

We use top-k mean average precision (MAP), mean average recall (MAR) and mean reciprocal rank (MRR) to evaluate the ranked lists at different time unit t. Rank k can be chosen to fit different applications. For example, considering that the rescue resources are often limited during a time window, we can predict hotspots that need the most help. In our case, Houston data has multiple ground-truth markers at one time unit (i.e., requests from multiple locations at the same time). We choose top-3, top-5, top-7, and top-10 MAP, MAR, and MRR in the experiment.

1. Mean Average Precision@K: We calculate average precision at top K of the predicted types at one time unit, then average over different time units. For computing average precision, if a predicted type matches any of the multiple ground-truth ones, we consider it a correct prediction.

2. Mean Average Recall@K: We calculate recall at top K of the predicted event types at one time unit and average over different time units.

3. Mean Reciprocal Rank@K: We compute the rank of each ground-truth types and the reciprocal rank score by $rr = \frac{1}{rank}$ and average over all test cases.

5.5 Results of testset likelihood In this experiment, for each synthetic dataset and real-world dataset, we create two different biased training sets using the strategies "Random" and "Top" as described in 5.2. We use the training set to train the model and evaluate the log-likelihood of the testset. The result of synthetic dataset and real-world dataset are shown in the Table 2 and Table 3 respectively.

For all synthetic and real-world datasets, our method $\text{DIL}_{\text{NN}}^{\text{E}}$ outperforms all the others in both biased strategies. Methods MLE_{H} and MLE_{R} are worse since they are MHPs and they do not explicitly model the biased generation process. MLE_{NN} is better since the neural network based model quantifies the influence of historical events in a more sophisticated way. Our model $\text{DIL}_{\text{NN}}^{\text{E}}$ outperforms MLE_{NN}, which indicates that the imitation framework with additional thinning process is more robust than MLE given biased training sequences.

Table 2: Testset log-likelihood comparison on two synthetic datasets with biased training sets selected by two different strategies "Random" and "Top".

Datasets	Bias Pattern	Methods			
		MLE_{H}	MLE_{R}	MLE_{NN}	$\text{DIL}_{\text{NN}}^{\text{E}}$
M-SYN	Rand	-5.6099	-5.1398	-3.3942	**-3.2726**
	Top	-9.7088	-8.0963	-4.0877	**-3.6943**
N-SYN	Rand	-3.0859	-3.5352	-1.3211	**-1.3004**
	Top	-5.6909	-6.0849	-2.4704	**-1.9654**

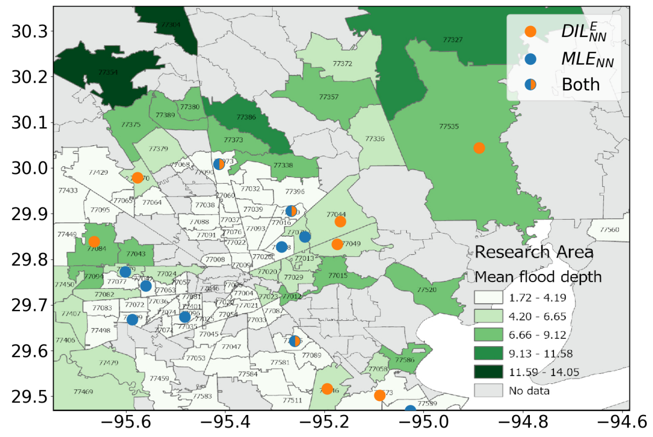
Table 3: Testset log-likelihood comparison on three real-world datasets with biased training sets selected by two different strategies "Random" and "Top".

Dataset	Bias Pattern	Methods			
		MLE_{H}	MLE_{R}	MLE_{NN}	$\text{DIL}_{\text{NN}}^{\text{E}}$
IPTV	Rand	-19.3445	-9.9676	-3.1611	**-3.1273**
	Top	-16.1876	-7.4249	-4.2732	**-4.1989**
Houston	Rand	-12.0878	-9.8246	-5.2327	**-5.1552**
	Top	-12.6778	-9.8425	-6.3368	**-5.9288**
Dallas	Rand	-15.3229	-10.3771	-9.0653	**-8.3042**
	Top	-11.0430	-10.6268	-8.2712	**-7.3074**

Furthermore, in the synthetic data experiment as shown in Table 2, for all models, the testing performance on the N-SYN datasets are better than the performance on the M-SYN datasets where the N-SYN dataset is generated by the neural self-modulating process. In general, for all methods, the testset results of "Top" are worse than those of "Random" since the imbalance rate of "Top" is higher.

5.6 Results of event type forecasting We evaluate the accuracy of predicted event types at test time based on historical events as described in Section 5.4. For Houston data, we use the first 21-hour data (about 70% events) as the training data and the last 5-hour data as the testing data. For Dallas data, we use the first 350-day data (about 70% events) as the training data and the last 30-day data as the testing data. For each dataset, we predict hotspots at each testing unit (e.g., an hours, a day), evaluate the top-K metrics described in Section 5.4, and average the performance over all units in the testing time window.

Table 4, 5 show prediction accuracy. We compare our models with the strongest competitor MLE_{NN} on Houston dataset . For almost all metrics evaluated at top K positions, our model $\text{DIL}_{\text{NN}}^{\text{E}}$ gets the best results. In particular, $\text{DIL}_{\text{NN}}^{\text{E}}$ using our proposed reward based on embeddings is better than $\text{DIL}_{\text{NN}}^{\text{W}}$ using traditional Wasserstein distance.


(a) Top-10 hotspots at test time 25 hour with "Random" biased training.

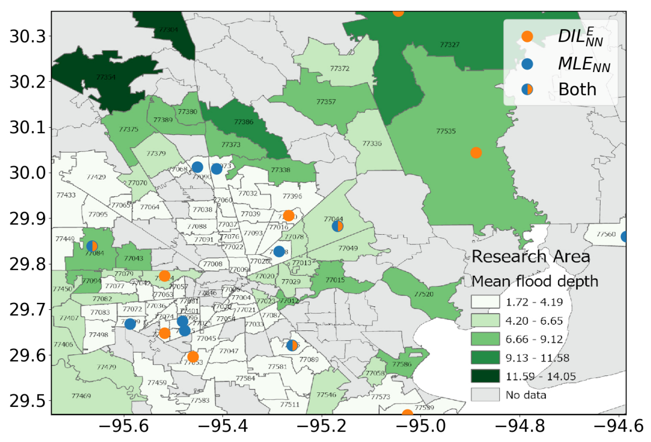

(b) Top-10 hotspots at test time 25 hour with "Top" biased training.

Figure 2: Houston map with "Mean flood depth" as the background. Blue dots and orange dots represent the predicted top ZIP codes using the strongest baseline $\mathrm{MLE_{NN}}$ and our method $\mathrm{DIL_{NN}^{E}}$, respectively.

Table 4: Hotspot detection in each testing hour for Houston rescue data where the training data is selected by "Random".

Method	Metrics	Top-K			
		Top-3	Top-5	Top-7	Top-10
$\mathrm{MLE_{NN}}$	MAP	**0.4111**	0.3707	0.3659	0.3477
	MAR	0.0257	0.0442	0.0745	0.0712
	MRR	0.1111	0.0867	0.0803	0.0667
$\mathrm{DIL_{NN}^{W}}$	MAP	0.3999	0.3720	0.3765	0.3504
	MAR	0.0385	0.0409	0.0633	0.0896
	MRR	0.1025	0.0933	0.0724	**0.0683**
$\mathrm{DIL_{NN}^{E}}$	MAP	0.3889	**0.3973**	**0.3947**	**0.3926**
	MAR	**0.0415**	**0.0494**	**0.0755**	**0.1037**
	MRR	**0.1333**	**0.1033**	**0.0884**	0.0681

Table 5: Hotspot detection in each testing hour for Houston rescue data where the training data is selected by "Top".

Method	Metrics	Top-K			
		Top-3	Top-5	Top-7	Top-10
$\mathrm{MLE_{NN}}$	MAP	0.3463	0.2911	0.3395	0.3082
	MAR	0.0570	0.0615	0.0662	0.0737
	MRR	0.0796	0.0716	0.0639	**0.0637**
$\mathrm{DIL_{NN}^{W}}$	MAP	0.3519	**0.4344**	0.3119	0.2565
	MAR	0.0662	0.0658	0.0703	0.0732
	MRR	0.0768	0.0667	0.0571	0.0578
$\mathrm{DIL_{NN}^{E}}$	MAP	**0.4130**	0.3811	**0.3602**	**0.3583**
	MAR	**0.0634**	**0.0654**	**0.0726**	**0.0801**
	MRR	**0.0811**	**0.0735**	**0.0733**	0.0612

5.7 Case study and visualization In Fig. 2, we visualize the predicted top-10 hotspots at 25^{th} hour on Houston rescue data based on two different biased training data. The blue dots represent the predicted top-10 ZIP codes from the strongest baseline $\mathrm{MLE_{NN}}$ and the orange dots shows the results of our method $\mathrm{DIL_{NN}^{E}}$. The dots of two colors indicate that the locations have been predicted as top-K regions by both methods. The background of the figures shows the map of Houston area with ZIP codes colored by "Mean flood depth". It is worth mentioning that "Mean flood depth" is computed based on historic flooding information and the areas with darker green in the backgrounds indicate higher flooding risks. During Hurricane Harvey, Harris County including ZIP codes 77044,77049, and 77084 is the worst-hit area. We can see that our method predicts more locations with higher flooding risks compared to the baseline. The prediction accuracy results compared to ground-truth are quantified in Table 4 and 5.

6 Conclusion

Real-world events often suffer from (unknown) biased observations due to external interference. Most of the existing work does not explicitly modeling the events with unknown biased generative mechanism and does not solve the mismatching problem between the distribution of training and the testing data caused by the sampling bias of external interference. We have proposed a novel DIL framework for learning unbiased TPPs from biased observations. The DIL framework makes the first attempt to learn TPPs modulated by additional thinning processes, which suppresses the model misspecification risk caused by the biased data. The DIL framework is applicable to arbitrary TPP models, whose reward function can be designed with high flexibility according to model architectures and application scenarios. In the future, we plan to consider more complicated additional thinning processes such as time varying sampling functions and introduce domain knowledge to the DIL framework.

7 Acknowledgement

This work was supported in part by the NSF under Grant No. 1927513, No. 1943486, No. 2147253, and NSF EPSCoR-Louisiana LAMDA program (Grant No. 1946231). The author would like to thank Kejin Wang for helping with Houston map images.

References

[1] G. Mohler, R. Raje, J. Carter, M. Valasik, and J. Brantingham, "A penalized likelihood method for balancing accuracy and fairness in predictive policing," in *Proc. of the IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 2018, pp. 2454–2459.

[2] H. Mei and J. M. Eisner, "The neural Hawkes process: A neurally self-modulating multivariate point process," in *Proc. of the Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2017, pp. 6754–6764.

[3] 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 *Proc. of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 2016, pp. 1555–1564.

[4] H. Xu, D. Luo, and H. Zha, "Learning Hawkes processes from short doubly-censored event sequences," in *Proc. of the International Conference on Machine Learning (ICML)*, 2017, pp. 3831–3840.

[5] C. R. Shelton, Z. Qin, and C. Shetty, "Hawkes process inference with missing data," in *Proc. of the AAAI Conference on Artificial Intelligence*, 2018.

[6] H. Mei, G. Qin, and J. Eisner, "Imputing missing events in continuous-time event streams," in *Proc. of the International Conference on Machine Learning*, 2019, pp. 4475–4485.

[7] A. M. Alaa, S. Hu, and M. Schaar, "Learning from clinical judgments: Semi-Markov-modulated marked Hawkes processes for risk prognosis," in *Proc. of the International Conference on Machine Learning (ICML)*, 2017, pp. 60–69.

[8] M. Norouzi, S. Bengio, N. Jaitly, M. Schuster, Y. Wu, D. Schuurmans *et al.*, "Reward augmented maximum likelihood for neural structured prediction," in *Proc. of the Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2016, pp. 1731–1739.

[9] Z. Zhou and M. Sun, "Multivariate Hawkes processes for incomplete biased data," in *Proc. of the IEEE International Conference on Big Data*, 2021, pp. 968–977.

[10] D. R. Cox and P. A. W. Lewis, "Multivariate point processes," in *Proc. 6th Berkeley Symp. Math. Statist. Prob*, vol. 3, 1972, pp. 401–448.

[11] H. Xu, M. Farajtabar, and H. Zha, "Learning granger causality for Hawkes processes," in *Proc. of the International Conference on Machine Learning (ICML)*, 2016, pp. 1717–1726.

[12] S. Zuo, H. Jiang, Z. Li, T. Zhao, and H. Zha, "Transformer Hawkes process," in *Proc. of the International Conference on Machine Learning (ICML)*, 2020, pp. 11 692–11 702.

[13] M. Farajtabar, N. Du, M. G. Rodriguez, I. Valera, H. Zha, and L. Song, "Shaping social activity by incentivizing users," in *Proc. of the Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2014, pp. 2474–2482.

[14] X. Guo, A. Hu, R. Xu, and J. Zhang, "Consistency and computation of regularized mles for multivariate hawkes processes," *arXiv preprint arXiv:1810.02955*, 2018.

[15] Y. Ogata, "On Lewis' simulation method for point processes," *IEEE Transactions on Information Theory*, vol. 27, no. 1, pp. 23–31, 1981.

[16] R. Zhang, C. Walder, and M.-A. Rizoiu, "Variational inference for sparse Gaussian process modulated Hawkes process," in *Proc. of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 04, 2020, pp. 6803–6810.

[17] S. Li, S. Xiao, S. Zhu, N. Du, Y. Xie, and L. Song, "Learning temporal point processes via reinforcement learning," in *Proc. of the Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2018.

[18] M. Eichler, R. Dahlhaus, and J. Dueck, "Graphical modeling for multivariate Hawkes processes with nonparametric link functions," *Journal of Time Series Analysis*, vol. 38, no. 2, pp. 225–242, 2017.

[19] S. Xiao, M. Farajtabar, X. Ye, J. Yan, L. Song, and H. Zha, "Wasserstein learning of deep generative point process models," in *Proc. of the Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2017, pp. 3250–3259.

[20] H. Xu, D. Luo, and H. Zha, "Hawkes processes on graphons," *arXiv preprint arXiv:2102.02741*, 2021.

[21] S. Zhu, S. Li, Z. Peng, and Y. Xie, "Imitation learning of neural spatio-temporal point processes," *IEEE Transactions on Knowledge and Data Engineering*, early access, 2021.

[22] W. Wu, J. Yan, X. Yang, and H. Zha, "Discovering temporal patterns for event sequence clustering via policy mixture model," *IEEE Transactions on Knowledge and Data Engineering*, 2020.

[23] J. Shang and M. Sun, "Geometric Hawkes processes with graph convolutional recurrent neural networks," in *Proc. of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 4878–4885.

[24] ——, "Local low-rank Hawkes processes for temporal user-item interactions," in *Proc. of the IEEE International Conference on Data Mining (ICDM)*, Nov. 2018, pp. 427–436.

[25] Q. Zhang, A. Lipani, O. Kirnap, and E. Yilmaz, "Self-attentive Hawkes process," in *Proc. of the International Conference on Machine Learning (ICML)*, 2020, pp. 11 183–11 193.

[26] Q. Zhang, A. Lipani, and E. Yilmaz, "Learning neural point processes with latent graphs," in *Proceedings of the Web Conference 2021*, 2021, pp. 1495–1505.

[27] V. Gupta, S. Bedathur, S. Bhattacharya, and A. De, "Learning temporal point processes with intermittent observations," in *Proc. of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2021, pp. 3790–3798.

[28] J. Shang, M. Sun, and N. S. Lam, "List-wise fairness criterion for point processes," in *Proc. of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 2020, pp. 1948–1958.

[29] U. Upadhyay, A. De, and M. Gomez-Rodriguez, "Deep reinforcement learning of marked temporal point processes," in *Proc. of the Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2018.